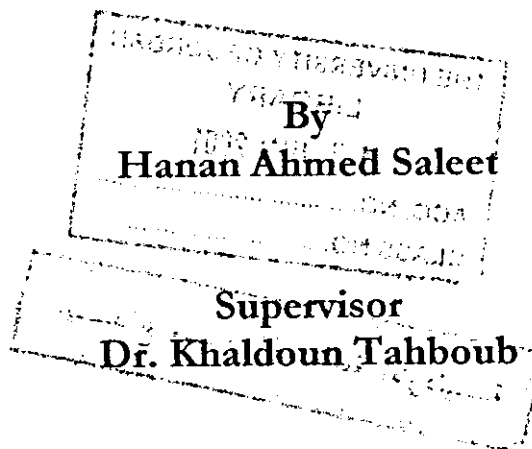


FLEXIBLE MANUFACTURING CELL FORMATION USING GENETIC ALGORITHMS

صه
م
ر
ا
ل
ت
= - =



تعتمد كلية الدراسات العليا
هذه النسخة من الرسالة
التوقيع: / التاريخ: /
C-1/17

Co- Supervisor
Prof. Dr. Ing. Dobrivoje Popovic

Submitted in Partial Fulfillment of the Requirements for
the Degree of Master of Science in
Industrial Engineering

Faculty of Graduate Studies
University of Jordan

June 2001

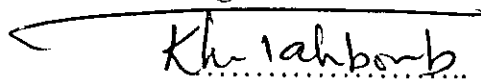
C-1/17
2
C.

This thesis was successfully defended and approved on 21st of May 2001.

Examination Committee

Dr. Khaldoun K. Tahboub, Supervisor
Industrial Engineering Department
University of Jordan

Signature

 Khaldoun K. Tahboub

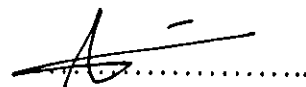
Dr. Zain El-Abideen Tahboub, Member
Industrial Engineering Department
University of Jordan

 Z. Tahboub

Dr. Mohammed Zaki Khader, Member
Electrical Engineering Department
University of Jordan

 M. Z. Khader

Dr. Adnan Mukattash, Member
Information Technology Department
Albalqa Applied University

 Adnan Mukattash

Dedication

To Abdulghafar, Whose Dreams Shaped My Life



To My Family, And My Dear Daughter Rana

Acknowledgement

I would like to acknowledge the help of my supervisor Dr. Khaldoun Tahboub who supported me with valuable ideas and resources to complete my thesis. Also I would like to thank Prof. Dr. Popovic' for his appreciated help and encouragement.

My deep appreciation for Dr. Deia' Abu-Annadi, University of Jordan, Electrical Engineering Department, for his advice and his valuable resources.

Table of contents

Committee Decision	ii
Dedication	iii
Acknowledgement	iv
Table of Contents	v
List of Tables	ix
List of Figures	xiii
Nomenclature	xiv
Abstract	xvii
1 <i>Introduction</i>	1
1.1 Introduction	1
1.2 Problem Statement and objectives	2
1.3 Organization of the Thesis	5
2 <i>Background and Literature Survey</i>	7
2.1 Introduction	7
2.2 Cellular Manufacturing	10
2.2.1 The manufacturing cell formation problem	11
2.3 Genetic Algorithms	14
2.3.1 Genetic algorithms definitions	14
2.3.2 Genetic algorithm versus traditional methods	14
2.3.3 Advantages of genetic algorithm	15
2.3.4 GA concept and methodology	16
2.3.4.1 similarity templates	17

2.3.4.2 GA mathematical foundation	18
2.3.5 Genetic algorithms approach	21
2.4 Literature Survey and Future Trends	25
2.4.1 Coarse-grained classification of grouping methods	25
2.4.1.1 Visual Inspection	25
2.4.1.2 Coding and Classification	25
2.4.1.3 Production Flow Analysis	26
2.4.2 Fine-grained classification of grouping methods	27
2.4.2.1 Array Based Methods	27
2.4.2.2 Hierarchical Clustering Methods	27
2.4.2.3 Graph-Based Approaches	28
2.4.2.4 Mathematical Programming Methods	28
2.4.2.5 Artificial Intelligence Techniques	29
2.5 Future Trends	32
2.6 Summary	33
3 Problem Formulation	34
3.1 Introduction	34
3.2 Initial Matrix (Data Set) Functions	36
3.3 Genetic Algorithm Functions	38
3.3.1 Initialization	39
3.3.2 Evaluation function	41
3.3.2.1 Grouping efficacy	42
3.3.2.2 Traveling Salesman Problem	44
3.3.3 Reproduction (Parent selection technique)	48

3.3.4 Crossover and mutation	49
3.3.4.1 Float representation	51
3.3.4.2 Order-based representation	53
3.3.5 Replacement	56
3.3.6 Termination	57
3.4 Experimental Results Graphical User Interface	58
3.4.1 Grouping efficiency	58
3.4.2 Cell index	60
3.4.3 Experimental results Graphical User Interface	62
3.5 Optimization of the Control Parameters for the Genetic Algorithm Using Metalevel GA	62
3.5.1 The space of the genetic algorithms	63
3.6 Summary	66
4 <i>Experimental Results</i>	67
4.1 The Programming Language: MATLAB	67
4.2 General Observations of the Metalevel Genetic Algorithm	69
4.3 Experimental Results	71
4.3.1 The importance of the cell index as a measure of the goodness of grouping solutions	77
4.3.2 Effects of constraining the number of permissible cells	78
4.3.3 Comparison of the GEC and the TSP results	81
4.4 Comparison with the Literature	82
4.5 Summary	83

5	<i>Summary, Conclusions and Recommendations</i>	84
5.1	Summary	84
5.2	Conclusions	85
5.3	Recommendations for Further Extensions	88
6	References	90
	Appendixes	94
Appendix A	MATLAB Software Description	94
	A.1 The Main Program: Graphical User Interface (GUI)	94
	A.2 The GAMCFPS Software Internal Functions	101
Appendix B	MATLAM Software Screens	106
Appendix C	Metalevel GA Solutions for the Ten Data Sets	112
	Abstract in Arabic	125

List of Tables

Table 1.1	0,1 incidence matrix	2
Table 1.2	The block diagonalized incidence matrix	3
Table 2.1	0,1 incidence matrix	12
Table 2.2	The block diagonalized incidence matrix	12
Table 2.3	The incidence matrix with intercell moves	13
Table 3.1	Benchmark test problems	38
Table 3.2	0,1 incidence matrix	39
Table 3.3	Initialization of GEC objective function	40
Table 3.4	Initialization of TSP objective function	41
Table 3.5	Intermediate solution	44
Table 3.6	Evaluation of GEC objective function.	44
Table 3.7	Distance matrix for parts	47
Table 3.8	Intermediate solution	47
Table 3.9	Evaluation of TSP objective function.	48
Table 3.10	The roulette wheel reproduction for the GEC objective function	50
Table 3.11	The roulette wheel reproduction for the TSP objective function	50
Table 3.12	Crossover and mutation in the GEC objective function	52
Table 3.13	Crossover and mutation in the TSP objective function	53
Table 3.14	The elitism replacement strategy for the GEC objective function	56
Table 3.15	The elitism replacement strategy for the TSP objective function	56
Table 4.1	Experiments Data Sets (Test Problems)	71
Table 4.2a	The experimentation results for the 10 sample data sets using simple GA	73

Table 4.2b	The experimentation results for the 10 sample data sets using metalevel GA	73
Table 4.3	The original incidence matrix for 15 x 10 Simple Chan and Milner problem (1982), data set number 6.	74
Table 4.4	Metalevel GA solution for the 15 x10 Simple Chan and Milner problem (1982), data set number 6	74
Table 4.5	The original incidence matrix for 20 x 35 Burbidge problem (1960), data set number 10	75
Table 4.6	Metalevel GA solution for the 20 x 35 Burbidge problem (1960), data set number 10. $\Gamma=75.8$ $\eta=88.1$ $CI=75.5$	76
Table 4.7	Alternative solutions with the same sum of voids and exceptions for the 14 x 24 King (1980c) problem	77
Table 4.8	Solutions to the Simple Chan and Milner (1982), problem number 6. When $k_{max} > k^*$ ($\Gamma=92.0$ $\eta=96.0$)	79
Table 4.9	Solutions to Complex Chan and Milner (1982), problem number 7. When $k_{max} > k^*$ ($\Gamma=85.4$ $\eta=94.5$)	79
Table 4.10	Comparison of the alternative configurations for Simple Chan and Milner problem when $kmax =2$	80
Table 4.11	Sample data set	81
Table 4.12	The solution matrix for the sample data set using GEC objective function. $eo=2$, $ev=1$, $\Gamma=83.3$, $\eta=93.9$, $CI=85.0$	81
Table 4.13	The solution matrix for the sample data set using TSP objective function. $eo=0$, $ev=9$, $\Gamma=65.4$, $\eta=82.7$, $CI=68.3$	81
Table C.1a	The original incidence matrix for the 6 x 10 King (1980a) problem, data set number 1	112

Table C.1b	The solution matrix for the 6 x 10 King (1980a) problem, data set 1. $\Gamma=72.4$, $\eta=86.3$, $CI=68.6$	112
Table C.2a	The original incidence matrix for the 6 x 15 Chen and Guerrero (1994) problem, data set number 2	113
Table C.2b	The near optimal solution matrix by GA for 6 x 15 Chen and Guerrero (1994) problem, data set 2. $\Gamma=61.7$, $\eta=80.1$, $CI=61.8$	113
Table C.2c	The optimal solution matrix for 6 x 15 Chen and Guerrero (1994) problem, data set 2. $\Gamma=64.2$, $CI=63.7$	113
Table C.3a	The original incidence matrix for the 9 x 9 Chu and Hayya (1991) problem, data set number 3	114
Table C.3b	The solution matrix for the 9 x 9 Chu and Hayya (1991) problem, data set number 3. $\Gamma=74.3$, $\eta=89.1$, $CI=70.4$	114
Table C.4a	The original incidence matrix for the 10 x 8 King (1980b) problem, data set number 4	115
Table C.4b	The solution matrix for the 10 x 8 King (1980b) problem, data set number 4. $\Gamma=82.2$, $\eta=92.5$, $CI=81.9$	115
Table C.5a	The original incidence matrix for the 10 x 12 Viswanathan (1996) problem, data set number 5	116
Table C.5b	The solution matrix for the 10 x 12 Viswanathan (1996) problem, data set number 5. $\Gamma=59.6$, $\eta=80.5$, $CI=58.3$	116
Table C.6a	The original incidence matrix for 15 x 10 simple Chan and Milner problem (1982), data set number 6.	117

542413

Table C.6b	Metalevel GA solution for the 15 x10 Simple Chan and Milner problem (1982), data set number 6. $\Gamma=92.0$, $\eta=96.0$, $CI=92.8$	117
Table C.7a	The original incidence matrix for the 15 x 10 Complex Chan and Milner problem (1982), data set number 7	118
Table C.7b	Metalevel GA solution for the 15 x10 Complex Chan and Milner problem (1982), data set number 7. $\Gamma=85.4$, $\eta=94.5$, $CI=86.1$	118
Table C.8a	The original incidence matrix for the 8 x 20 Chandrasekharan and Rajagopalan (1986) problem, data set number 8.	119
Table C.8b	The solution matrix for for the 8 x 20 Chandrasekharan and Rajagopalan (1986) problem, data set number 8. $\Gamma=85.2$, $\eta=95.8$, $CI=84.4$	119
Table C.9a	The original incidence matrix for 14 x 24 King (1980) problem, data set number 9	120
Table C.9b	the first solution matrix for the 14 x 24 King (1980b)problem, data set 9. $\Gamma=61.4$, $\eta=81.3$, $CI=61.2$	121
Table C.9c	the second solution matrix for the 14 x 24 King (1980b)problem, data set 9. $\Gamma=61.8$, $\eta=81.4$, $CI=62.4$	122
Table C.10a	The original incidence matrix for 20 x 35 Burbidge problem (1960), data set number 10	123
Table C.10b	Metalevel GA solution for the 20 x 35 Burbidge problem (1960), data set number 10. $\Gamma=75.8$ $\eta=88.1$ $CI=75.5$	124

List of Figures

Figure 2.1	The roulette wheel	22
Figure 3.1	The main menu of MATLAB software	35
Figure 3.2	The simple GA flow diagram	37
Figure 4.1	The best, average and standard deviation variation with the number of generations.	72
Figure 4.2	Average number of generations versus selected values of k_{max}	80
Figure A.1	The main menu of MATLAB software	96
Figure A.2	The metalevel GA flow diagram	99
Figure A.3	The second level GA flow diagram	100
Figure A.4	The main functions of the MATLAB software	102
Figure B.1	The main menu of the MATLAB software	106
Figure B.2	The second level GA GUI	107
Figure B.3	The simple GA GUI	108
Figure B.4	The results GUI	109
Figure B.5	The help GUI	110
Figure B.6	Plot of the fitness values with the number of generations	111

Nomenclature

Γ	Grouping Efficacy
η	Grouping Efficiency
φ	The ratio of the number of exceptional elements to the total number of operations
ϕ	The ratio of the number of voids in the diagonal blocks to the total number of operations
$\delta(H)$	Defining length of a schema
$\rho(i)$	Permutation of machines
$\sigma(j)$	Permutation of parts
η_1	The ratio of the number of 1's in the diagonal blocks
η_2	The ratio of the number of 1's in the off diagonal blocks
$\sum f_i$	Total fitness for all solutions
$A [n,m]$	$m \times n$ (0,1) incidence matrix
AI	Artificial Intelligence
A_i	String number i
a_{ij}	$a_{ij}=1$, if part j needs operation on machine i ; $a_{ij}=0$, otherwise
BDF	Block Diagonal Form
C	Crossover rate
CI	Cell Index
CPU time	Central Processing Unit time
d_{ij}	Distance (similarity) measure between a pair of columns (rows) i and j
c	Total number of ones (operations) in the incidence matrix
$E\{\text{count}_i\}$	The expected count of string i (chromosome i)

e_d	Number of ones inside the diagonal blocks
e_o	Number of exceptional elements
e_v	Number of voids
f	Average fitness
FFA	Factory Flow Analysis
f_i	Fitness value of string A_i
FMS	Flexible Manufacturing System
GA	Genetic Algorithm
GAERL.m	Genetic algorithm experimental results interface, which is a MATLAB GUI
GAMCFPS	Genetic Algorithm Manufacturing Cell Formation Problem Solver
GAMCFPS	Genetic Algorithm Manufacturing Cell Formation Problem
GAs	The standard Genetic Algorithm
GEC	Grouping Efficacy
GT	Group Technology
GUI	Graphical User Interface
H	A schema
k^*	The optimal number of cells
k^L	The best known number of cells from the literature
K_{max}	Maximum number of cell specified by the user
M	Mutation rate
m	Number of machines in the (0,1) incidence matrix
MATLAB	MATrix LABoratory
MCFL.m	Main Cell Formation Interface, which is a MATLAB GUI
MCFP	Manufacturing Cell Formation Problem

MHC	Modified Hamiltonian Chain
n	Number of parts in the (0,1) incidence matrix
N	Population size
NM	Number of bits to undergo mutation
O(H)	The fixed positions within the schema
OSHN	Ortho-Synapse Hopfield Network
P select i	Probability of selection of ith chromosome
Pc	Crossover Survival Probability (rate)
PFA	Production Flow Analysis
Pi	Selection Probability of String
Pm	Mutation Probability (rate)
PMX	Partially Mapped Crossover
Ps	Survival Probability
q	Weighing factor of grouping efficiency (0-1)
SLGAI.m	Second level GA interface which is a MATLAB, GUI
SMGAI.m	Simple GA interface which is a MATLAB GUI
t	Time step
TSD	Target Standard Deviation
TSP	Travelling Salesman Problem

Abstract

FLEXIBLE MANUFACTURING CELL FORMATION USING GENETIC ALGORITHMS

By
Hanan Ahmed Saleet

Supervisor
Dr. Khaldoun Tahboub

Co- Supervisor
Professor Dr. Ing. Dobrivoje Popovic

Industries are embracing the concepts of agile manufacturing, flexible manufacturing and group technology, which favor nimble principles over the aging techniques of mass production. These advanced manufacturing concepts are characterized by their ability to allow rapid response to continuously changing customer requirements. Cellular manufacturing is the implementation of group technology (GT) to the manufacturing process. The aim of cellular manufacturing is to decompose the manufacturing process into a number of machine cells, which are dedicated to the production of corresponding part families.

This research presents a method of using a genetic algorithm(GA) to form machine cells and part families with minimum intercellular movements i.e. minimum number of exceptional elements and minimum number of voids. In order to solve the manufacturing cell formation problem a metalevel genetic algorithm is constructed to analyze data sets taken from the literature so that Block Diagonal Forms are obtained to achieve minimum intercellular movements. The GA methodology can find near-optimal solutions by using the roulette wheel reproduction approach, a combination of crossover techniques, a combination of mutation techniques, elitism replacement and hybrid

termination strategies. In addition it uses two objective functions. The first is grouping efficacy, which attempts to minimize the number of voids plus the number of exceptional elements divided by the total operational zone. The operational zone is defined by the number of operations (exceptional elements plus ones along the diagonals). The other is the Travelling Salesman Problem formulation which attempts to determine the desirable permutation for rows and columns in a solution matrix by using a distance (similarity) measure between a pair of rows (machines) and columns (parts).

The major contributions of this thesis may be stated as:

1. This cell formation methodology offers improved flexibility since it allows the cell designer to use the grouping efficacy or the travelling salesman problem formulation as objective functions and to incorporate design constraints during cell formation. These capabilities allow alternative cell configuration to be generated and reviewed easily.
2. Using metalevel GA where the control parameters i.e. population size, target standard deviation, mutation rate and crossover rates are determined by the first level GA and used to solve the second level GA, which gives the cell formation problem solutions.
3. A MATLAB software is constructed to handle the whole process of solving the cell formation problem, by controlling and monitoring each step in generating near-optimal solution.
4. A comparative study has been conducted between the GA solutions and the benchmark solutions to the problems adopted from the literature. This comparison is done using grouping measures such as grouping efficiency, grouping efficacy and cell index. The results obtained are at least equal to any previously reported results.

1- INTRODUCTION

1.1 Introduction

Many firms have adopted Flexible Manufacturing Systems (FMSs) as a means to produce high quality products with small lead times in order to meet the growing requirements of customized production. The development of FMS is mainly concerned with achieving productivity in a well-balanced transfer line, while retaining the flexibility of a low volume job-shop-type conventional manufacturing system. There are a variety of problems to be addressed for the successful development and implementation of an FMS (Tiwari and Vidyarthi, 2000). The stepping stone for the implementation of an FMS is considered to be cellular manufacturing. Cellular manufacturing is an implementation of group technology (GT) to the manufacturing process.

The primary input data are derived from route sheets. This data is in the form of a zero-one matrix where the rows represent the machines and the columns represent the parts. An element a_{ij} of the matrix is one if the j th component visits the i th machine for processing; otherwise it is zero. Algorithms that aim at forming the part families and machine cells essentially try to rearrange the rows and columns of the matrix to get a block diagonal form. The ideal situation is one in which all the ones are in the diagonal blocks and all the zeros are in the off-diagonal blocks. However, the ideal case seldom occurs in a real shop floor problem. The block diagonal form is usually far from perfect. This could be either due to the properties of the data or the inadequacies of the algorithm or both.

A complete block diagonal matrix, in which mutually independent machine-component groups can be identified, is ideal for the successful development of a cellular manufacturing system. As the number of parts requiring operations in more than one machine cell (exceptional parts) increases, the effectiveness of the corresponding cellular

manufacturing system decreases. This is due to intercellular material handling costs associated with exceptional parts and the necessary adjustments in the cellular manufacturing system to accommodate the processing of these exceptional parts.

The problem of identifying machine cells and the corresponding part families in cellular manufacturing has been extensively researched over the last thirty years. However, the complexity of the problem and the considerable number of issues involved in its solution create the need for increasingly efficient algorithms. In this thesis the use of genetic algorithms for the solution of a simple version of the problem is investigated.

Unlike many other optimization techniques, GA does not make strong assumptions about the form of the objective function. Most optimization methods maintain a single solution and improve it until an optimal solution is found. GA differs in that it maintains and manipulates a family, or a population of solutions, in the search for an optimal solution. GA mimics the evolutionary process by implementing a 'survival of the fittest' strategy. In general, the fittest individuals of any population tend to reproduce and pass their genes to the next generation, thus improving successive generations. However, some of the worst individuals do, by chance, survive and reproduce (Goldberg,1989).

1.2 Problem Statement and Objectives

As stated earlier, the solution to the cell formation problem is obtained by forming blocks along the diagonals of a (0,1) incidence matrix shown in Table 1.1. and Table 1.2.

Table 1.1: 0,1 incidence matrix

	p1	p2	p3	p4	p5
m1	1	1	1	0	1
m2	1	0	0	1	0
m3	0	1	0	0	1

Table 1.2: The block diagonalized incidence matrix

	p4	p1	p5	p3	p2
m2	1	1	0	0	0
m1	0	1	1	1	1
m3	0	0	1	0	1

The cell formation problem can be solved by forming a set of completely autonomous machine cells that minimizes the intercellular movements, by minimizing the number of exceptional elements e.g., element (p1, m1) in Table 1.2 and the number of voids e.g., element (p3,m3) in Table 1.2.

In this thesis, GA technique is used to solve the cell formation problem. A MATLAB software is constructed utilizing GA approach to solve the manufacturing cell formation problem. The software is constructed to be a comprehensive computer package to handle the whole process by controlling and monitoring each step of generating near-optimal solutions. The main features of the developed software are:

1. The use of metalevel GA where the control parameters i.e., population size, target standard deviation, crossover rate and mutation rate are determined by the first level GA and are used to solve the second level GA, which gives the cell formation problem solutions.
2. The user is given the option to use the simple GA after specifying the previous parameters.
3. The user is given the option to switch between two objective functions; Grouping Efficacy (GEC) or Travelling Salesman Problem formulation (TSP).
4. The specification of the maximum number of cells (k_{max}) so that, the solved matrix may be partitioned into a number of cells less or equal to this number

Data sets taken from the literature which represent the (0, 1) incidence matrices are the main input to the genetic algorithm software, which uses roulette wheel selection, a combination of crossover techniques, a combination of mutation techniques, elitism replacement strategy and hybrid termination criteria. This software contains several graphical user interfaces (GUI) and several functions.

To increase the flexibility of the proposed algorithm the user is allowed to incorporate or selectively remove constraints on the number of permissible cells. Unconstrained solutions containing the naturally occurring clusters can be generated as well as constrained solutions. By allowing alternative cell configurations evaluation, the power of the technique as a tool of analysis is extended.

A separate graphical user interface is constructed to handle the outputs and to display the results including grouping efficacy, grouping efficiency, TSP results, number of voids, number of exceptional elements, number of cells and the number of iterations required to find near optimal solution. Also it displays a plot of the maximum, the average and the standard deviation of the fitness values variation with the number of generations.

The objectives of the research are as follows:

- a. Apply the simple genetic algorithm to solve the manufacturing cell formation problem (MCFP), by setting the GA control parameters randomly.
- b. Apply a metalevel GA to the MCFP. The first level GA optimizes the control parameters used in the second level GA in a trail to find the near optimal solution.
- c. Formulation and implementation of genetic algorithm, that can give the near optimal solutions to the manufacturing cell formation problem, allowing the cell designer to

use various objective functions and incorporate design constraints during cell formation.

- d. Constructing a comprehensive user-interface software to handle the whole evolutionary process, by controlling and monitoring each step of generating near-optimal solution.
- e. Comparing the GA results with the benchmark solutions to the problems adopted from the literature.

1.3 Organization of the Thesis

The thesis is organized as follows:

Chapter one is an introduction, it contains a general introduction about manufacturing cell formation problem (MCFP), genetic algorithms, and the application of GA to solve the MCFP, and the problem statement and objectives.

Chapter two provides background and literature survey. It contains an introduction to the manufacturing cell formation and its importance in the group technology. Genetic algorithm and its implementation to cell formation are discussed. This chapter gives an overview of genetic algorithms, i.e., GA basic theory, GA concept, definitions, advantages, and the differences from traditional methods. It also shows the similarity templates (schemata) and the GA mathematical foundations. In addition it describes GA approach and methodology.

Chapter three represents the core of the thesis since it provides a complete description of the problem formulation and solution methodology for manufacturing cell formation using genetic algorithms. The manual formulation of the computer implementation is described here. Finally, it presents the methodology of the metalevel GA.

Chapter four gives an introduction to MATLAB, the computer language used in this thesis to build a software with flexible characteristics. Also it presents a comparison of the results of applying the simple genetic algorithm and the metalevel GA. Constraining the number of cells and the different measures for the goodness of grouping are discussed in this chapter. Finally, a comparison with the literature is presented.

Chapter five contains summary, conclusions and recommendations.

2- BACKGROUND AND LITERATURE SURVEY

2.1 Introduction

Group Technology (GT) has been recognized as an important disciplined approach to low volume/high-variety and mid-variety manufacturing, which included 50-75% of manufactured parts and is likely to increase. Cellular manufacturing, based on the philosophy of GT, is intended to make small-to medium-sized batches of a large variety of part types produced in the flow shop manner. One of the most important problems in designing a cellular manufacturing system is the formation of machine cells and the grouping of part families (Chunwei and Zhiming, 2000).

Machine-cell formation algorithms try to block-diagonalize the machine-part incidence matrix into a block diagonal form in which all '1' entries are located in the diagonal blocks and all the '0' entries are located in the off diagonal blocks in an ideal case.

In general, most cell formation techniques work well for small, well-defined problems. Although some methods offer generally superior results, no single technique has been shown to provide the best solution for a broad range of applications. Some techniques do not offer cell designers the flexibility to change objective functions and selectively include constraints. In addition, several meaningful evaluation criteria have been proposed that cannot explicitly be used as an objective function in many clustering algorithms. Finally, some clustering algorithms cannot identify all naturally occurring clusters and find solutions with a constrained number of clusters (Jeffrey *et al.*, 1996).

Cell formation approaches in the literature have covered a wide spectrum of techniques. Array-based clustering methods perform a series of column and row manipulations to form part families and machine cells simultaneously. The Bond Energy Algorithm and

the Rank Order Clustering Algorithm are two of the oldest and commonly array-based methods.

Graph-based methods employ a graph or network representation of the cell formation problem and use partitioning techniques, such as the Modified Hamiltonian Chain (MHC) to create manufacturing cells.

Neural networks techniques were also used to identify machine cells and part families. The networks range from simple nets to greatly modified nets such as Ortho-Synapse Hopfield Network (OSHN) which solves large incidence matrices in a very short time.

Also, the genetic algorithms are used to solve the cell formation problem which uses different objective functions, such as minimizing the intercellular movements, minimizing costs due to intercell and intracell part movements and minimizing the total within cell load variation. In addition, genetic programming which evolve variable-length strings in the form of computer programs, is used to give solutions to the cell formation problem.

To meet the need for a flexible, efficient, and effective clustering technique, we have developed a cell formation method that uses genetic algorithm. GA can find solutions to linear and nonlinear problems by simultaneously exploring multiple regions of the state space and exponentially exploiting promising areas through mutation, crossover and selection operations. Genetic algorithms have proved to be an effective and flexible optimization tool that can find optimal or near-optimal solutions.

Some difficulties can be faced when solving the cell formation problem, such as:

1. Achieving the data sets to verify the proposed algorithm
2. GA manipulation and formulation.
3. GA parameters optimization.
4. Converging to the optimal solution.
5. Building a flexible and a comprehensive simulator to handle all the previous problems.

The manipulation and formulation of the GA program required a great effort. But the final results were very efficient, especially, the achievement of optimized GA parameters, which give better solutions to the cell formation problem.

Genetic algorithms are significantly different from alternative search strategies. The benefits of using GA to solve the cell formation problem are summarize as follows (Goldberge, 1989):

1. The mechanics of GA are surprisingly simple, involving nothing more than copying strings (solutions) and swapping partial strings.
2. Rather than using the parameters themselves, GA uses a coding of the parameters. From an artificial intelligence perspective, a GA is subsymbolic, representing and applying knowledge without symbol manipulation.
3. GA search from a population of points rather than from a single point. GA arrives at a result by improving on the best solutions over a number of generations, so GA will reduce the computer processing time.
4. They use payoff information, not derivatives or other auxiliary knowledge. GA can be successful even when response surfaces are multi-objective, discontinuous and non-differentiable.

5. Genetic algorithms make use of probabilistic rather than deterministic transition rules, random choice is used to explore regions of the search space that are likely to contain improvements.
6. GA is a robust and flexible optimization technique, so further modifications to the existing solution can be done easily. GA parameters affect the optimality of the final solution and the CPU time.

The cell formation problem, the genetic algorithms and the literature survey are discussed in the following sections. For the knowledgeable reader, not going through the following sections will not affect the other chapters.

2.2 Cellular Manufacturing

Cellular manufacturing is the implementation of group technology (GT) to the manufacturing process. GT was originally introduced by a Russian engineer, S. P. Mitrovanov in 1958 (Vajpayee, 1995) and was popularized in the west by Burbidge in 1975. The basic idea of group technology (GT) is to decompose a manufacturing system into subsystem. It states that significant advantages can be achieved by grouping company elements that are bound by some form of similarity. These elements range from entire departmental units to simple machines or parts. On the plant level the aim of GT is to decompose the manufacturing process into a number of machine cells which are dedicated to the production of corresponding part families. This configuration is traditionally known as cellular manufacturing. The intuition behind cellular manufacturing is an attempt to achieve the mass-production effect of flow-line production in batch manufacturing. The implementation of cellular manufacturing has been reported to result in significant benefits for the manufacturing procedure.

Some of these benefits are (Dimopoulos and Mort, 2001):

- Reduced set up times ;
- Reduced work-in-progress inventory ;
- Reduced throughput times;
- Reduced material handling costs;
- Simplified scheduling;
- Simplified flow of products;
- Improved quality.

2.2.1 The manufacturing cell formation problem

the MCFP can be considered as multi-criteria decision-making problem. Objectives such as minimization of costs of intercellular movements, space usage, part subcontracting, machine loading and operation costs, or the minimization of the amount of intercellular movements, part dissimilarities and machine duplications can be formulated to solve the MCFP (Mansouri et al., 2000). Most researchers when dealing with cell formation they assume an infinite loading capacity. But from practical point of view it is important to incorporate machine loading into the model. Also, To achieve higher level of production flexibility, it becomes necessary to consider alternative process plans for each part.

Although these objectives consider many manufacturing factors, they are often difficult to formulate for practitioners. So the simple binary version of the cell-formation problem is considered in this thesis where the objective is to minimize the intercellular movements may provide rough cut design. It is usually illustrated with the help of the machine-component incidence matrix $A [m \times n]$ where:

m is the total number of machines in the plant,

n is the total number of parts in the plant.

Each position in the matrix can assume two values, '0' and '1'. A positive entry indicates that the part of the corresponding column has an operation on the machine of the corresponding row. A '0' entry indicates the opposite. The information provided by the incidence matrix is illustrated with a simple case of a plant that produces five parts using three machines. By analyzing information from the route cards of parts, the incidence matrix of Table 2.1 is obtained. The value of $a_{2,4}$ is equal to '1' thus part 4 needs an operation on machine 2. In contrast, part 4 does not need an operation on machine 1 since $a_{1,4}$ is equal to '0'.

Table 2.1: 0,1 incidence matrix

	p1	p2	p3	p4	p5
m1	0	1	1	0	1
m2	1	0	0	1	0
m3	0	1	1	0	1

Once the incidence matrix has been obtained, the cell-formation problem is transformed to the problem of finding a configuration with all positive entries arranged inside blocks along the main diagonal of the incidence matrix. A diagonalized matrix allows the easy identification of machine cells and corresponding part families. Table 2.2 illustrates the diagonalized version of the example matrix, which results from rearranging its rows and columns.

Table 2.2: The block diagonalized incidence matrix

	p4	p1	p5	p3	p2
m2	1	1	0	0	0
m1	0	0	1	1	1
m3	0	0	1	1	1

By observing the matrix it is easy to identify two independent cells, the first one comprising of machine 2 and parts 1 and 4, and the second one comprising of machines 1 and 3 and parts 2, 3 and 5. The main objective of a cell-formation algorithm in the simple version of the problem is the construction of completely independent cells, i.e.

cells where all parts included in a part family are solely processed within the corresponding machine cell. However, this is a case rarely encountered in practice. Table 2.3 illustrates a situation where the cells that have been formed are not independent.

Table 2.3: The incidence matrix with intercell moves.

	p4	p1	p5	p3	p2
m2	1	1	0	0	0
m1	0	1	1	1	1
m3	0	0	1	0	1

The reason for this inefficiency is part 1, which requires an operation on a machine that belongs to a different cell (machine 1). It is customary in cellular manufacturing terminology to describe part 1 as an exceptional part or overlapping part and machine 1 as a bottleneck machine. When completely independent cells cannot be formed, the objective usually becomes the minimization of intercell moves or the minimization of material handling costs in general.

So, exceptional elements are machines or parts which do not permit the arrangement of the incidence matrix in a strictly block diagonal form. In other words, they cause machines to be shared by different families of parts. Element $a_{3,3}$ in Table 2.3 is called a void (zero inside the cell) indicates that a machine assigned to a cell is not required for the processing of a part in the cell. The implications of voids and exceptions are as follows:

1. Control and co-ordinate efforts. Cell formation aims to achieve small independent cells. Small cells are easy to control. When the cells are not independent co-ordination is required for processing part operation in different cells.

2. Intracell and intercell material handling. The presence of voids may cause increase in intracell handling and decrease the machines utilization, and the presence of exceptions will lead to intercell handling.

2.3 Genetic Algorithm

2.3.1 Genetic algorithm definitions

John Holland introduced GA in 1975. It has been applied to a number of fields like mathematics, engineering, biology etc.

Genetic Algorithm can be defined as follows:

- genetic algorithm: is search algorithm based on the mechanics of natural selection and natural genetics. It combines 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 (Goldberg, 1989).
- genetic algorithms: is stochastic technique for solving combinatorial optimization problems, it finds solutions to linear and non-linear problems by simultaneously exploring multiple regions of the state space and exponentially exploiting promising areas through mutation, crossover, an selection operations.

2.3.2 Genetic algorithm versus traditional methods

In order for the genetic algorithm to surpass its more traditional cousins in the quest for robustness, GA must differ in some very fundamental ways. Genetic algorithm is different from more normal optimization and search procedures in many ways:

1. GA works with a coding of the parameter set not the parameters themselves.
2. GA searches from a population of points, not a single point. GA operates on several solutions simultaneously, gathering information from current search

points to direct subsequent search. The ability to maintain multiple solutions concurrently makes GA less susceptible to the problems of local optima.

3. GA uses payoff (objective function) information, not derivatives or other auxiliary knowledge.
4. GA uses probabilistic transition rules, not deterministic rules. GA is randomized algorithm, in that it uses operators whose results are governed by probability. These results for such operations are based on the value of a random number.
5. GA optimizes the trade-off between exploring new points in the search space and exploiting the information discovered thus far.
6. GA has the property of implicit parallelism. Implicit parallelism means that the GA's effect is equivalent to an extensive search of hyperplanes of the given space, without directly testing all hyperplane values.

2.3.3 Advantages of genetic algorithm

The advantages of using genetic research and optimization are summarized as follows (Goldberg, 1989):

1. The GA is computationally simple yet powerful in its search for improvement. Furthermore it is not limited by restrictive assumptions about the search space such as: assumptions concerning continuity, existence of derivatives and other matters.
2. The search is conducted from a population of points rather than a single point, thus increasing the exploratory capability of GA.
3. GA is theoretically and empirically proven to provide robust search in complex spaces. If artificial systems can be made more robust, costly redesign can be reduced or eliminated.

4. GA works with a direct coding of the parameter set rather than the parameters themselves so, it is suitable for discontinuous, high dimensional, and multi-modal problems.
5. GA is example on search procedures that use random choice (probabilistic transition rules) as a tool to guide a highly exploitative search through a coding of a parameter space.
6. GA has no need for any auxiliary information: GA is blind. To perform an effective search for better structures it only need payoff values (objective function values) associated with individual strings (solutions).
7. GA lends itself naturally to implementation in parallel processing environment leading to the ability to exploit newer technologies in this domain, so faster computational times. GA is now finding more widespread application in business, scientific and engineering.
8. GA has certain input parameters, which can be varied to improve on poor performance such as initial population size, number of reproductions per generation, mutation rate etc.

2.3.4 GA concept and methodology

Genetic algorithms belong to the family of evolutionary algorithms, a research field that has expanded rapidly over the last ten years. Evolutionary computation employs the concept of Darwinian strife for survival to guide the search for a potential solution. The probabilistic nature of evolutionary algorithms and their ability to search in parallel through the solutions' space means that they are less likely to be trapped in local optima. The simple GA is composed of three operators: reproduction, crossover and mutation.

Reproduction is a process in which individual strings are copied according to their objective function values. Copying strings according to their fitness values means that strings with a higher value have a higher probability of contributing one or more offspring (child) in the next generation.

After reproduction, the selected (copied) strings enter into a 'mating pool', a term given to the place where the selected strings are stored before the crossover operator is applied. Each string in the mating pool randomly selects its partner to participate in crossover. The exact locations of the crossovers are determined randomly.

Reproduction and crossover are usually far more important to the GA than mutation. Mutation can protect against losses that could be irrevocable by randomly changing the value of a string position. It can also cause local or global movement in the search space in effect, giving the search a new place to explore. Mutation probabilities are usually set very low. Mutation operator is a random alteration of a string, such as switching a bit.

2.3.4.1 Similarity templates

A schema is a similarity template describing a subset of strings with similarities at certain string positions. The idea of schema provides a powerful and compact way to talk about all the well-defined similarities among finite-length strings over a finite alphabet (Samhuri, 2000). It is recognized that if the strings are considered separately, then few pieces of information will be available, however, when the strings, their fitness values, and the similarities among the strings in the population are considered, a wealth of new information to help direct the search is admitted. A binary schemata consist of three alphabets: $\{*, 0, 1\}$ where * is a do not care symbol, so it can take a value of 0 or 1.

In general, a particular string contains 2^L schemata, where L is the string length. As a result, a population of size n contains somewhere between 2^L and $(n \cdot 2^L)$ schemata depending upon the population diversity. The original motivation for considering important similarities was to get more information to help guide the search.

The effect of reproduction on a particular schema is easy to determine, since most fit strings, have higher probabilities of selection. Hence, reproduction will not sample new points in the space. Crossover leaves a schema unchanged if it does not cut the schema, but it disrupt a schema when it does. Mutation at normal low rates does not disrupt a particular schema very frequently.

2.3.4.2 GA mathematical foundation

The following mathematical foundation can be found in details in Goldberg (1989):

The order of a scheme H , denoted by $o(H)$, is simply the number of fixed position (in a binary alphabet, the number of 1's and 0's) present in the template. The defining length of a schema H , denoted by $\delta(H)$, is the distance between the first and last specific string position.

Suppose at a given time step there are m examples of a particular schema H contained within the population $\Lambda(t)$ where we write $m = m(H, t)$. During reproduction, a string is copied according to its fitness, or more precisely a string A , gets selected with probability $p_i = f_i / \sum f_i$. After picking a population of size n from the population $\Lambda(t)$, we expect to have $m(H, t+1)$ representatives of the schema H in the population at time $t+1$ as given by the equation $m(H, t+1) = m(H, t) \cdot n \cdot f(H) / \sum f_i$ where $f(H)$ is the average fitness of the strings representing schema H at time t . If the average fitness of the entire population is

written as $\bar{f} = \sum f_i / n$, then the reproductive schema growth equation is rewrite as follows:

$$m(H, t) = m(H, t + 1) \cdot \frac{f(H)}{\bar{f}} \dots\dots\dots(2.1)$$

In words, a particular schema grows as the ratio of the average fitness of the schema to the average fitness of the population.

Suppose that a particular schema H remains above average an amount cf with c a constant. Equation (1) can be written as follows:

$$m(H, t + 1) = m(H, t) \cdot \frac{(\bar{f} + cf)}{\bar{f}} = (1 + c) \cdot m(H, t) \dots\dots\dots(2.2)$$

starting at $t = 0$ and assuming a stationary value of c, equation(2) will be as follows:

$$m(H, t) = m(H, 0) \cdot (1 + c)^t \dots\dots\dots(2.3)$$

A lower bound on crossover survival probability P_s can be calculated for any schema. Because a schema survives when the cross site falls outside the defining length, the survival probability under simple crossover is $P_s = 1 - \delta(H) / (L - 1)$. If crossover itself is performed by random choice, say with probability P_c at a particular mating, the survival probability may be given by the expression:

$$P_s \geq \left[1 - \left(P_c \cdot \frac{\delta(H)}{L - 1} \right) \right] \dots\dots\dots(2.4)$$

The combined effect of production and crossover may now be considered. The number of a particular schema H expected in the next generation is as follows:

$$m(H, t + 1) \geq m(H, t) \cdot \frac{f(H)}{f} \cdot P_s \dots\dots\dots(2.5)$$

$$m(H, t + 1) \geq m(H, t) \cdot \frac{f(H)}{f} \left[1 - \left(P_c \cdot \frac{\delta(H)}{L-1} \right) \right] \dots\dots\dots(2.6)$$

With reproduction and crossover, schema H grows or decays depending upon a multiplication factor. With both crossover and reproduction, that factor depends on two things: whether the schema is above or below the population average and whether the schema has relatively short or long defining length. Clearly, those schemata with both above-average observed performance and short defining lengths are going to be sampled at increasing rates.

Mutation is the random alteration of a single position with probability P_m . In order for a schema H to survive, all of the specified positions must themselves survive. Therefore, since a single bit survives with probability $(1-P_m)$, and since each of the mutations is statistically independent, a particular schema survives when each of the $o(H)$ fixed positions within the schema survives. Multiplying the survival probability $(1-P_m)$ by itself $o(H)$ times, we obtain the probability of surviving mutation, $(1-P_m)^{o(H)}$. For small values of P_m ($P_m \ll 1$), the schema survival probability may be approximated by the expression $1 - o(H) \cdot P_m$. So, a particular schema H receives an expected number of copies in the next generation under reproduction, crossover, and mutation as given by the following equation (ignoring small cross-product terms):

$$m(H, t + 1) \geq m(H, t) \cdot \frac{f(H)}{f} \left[1 - \left(P_c \cdot \frac{\delta(H)}{L-1} \right) - O(H)P_m \right] \dots\dots\dots(2.7)$$

This equation indicates that: short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations. This conclusion is so important

that it has the special name of **Schema Theorem**, or the **Fundamental Theorem of Genetic Algorithms**.

2.3.5 Genetic algorithms approach

The GA search is conducted from a population of solutions, each solution or individual in the population is described by a vector of variables (chromosome representation).

The first step in the GA procedure is to initialize the population either randomly or by seeding. Once the initial population is generated, each individual i , is evaluated by using the objective function to determine its fitness or value, f_i . A subset of the population is selected to parent the next generation, where the selection mechanism takes note of diversity among individuals as well as of individual performance (Wenston, 1993). An individual in the population can be selected to be a parent more than once. A probabilistic selection is performed such that the fittest individuals have an increased chance of being selected.

These parents then undergo reproduction using genetic operators to produce a new population. To complete the new population, a subset of the old population is added to the new population. For example, the elitist model ensures that the best individual of one generation is included in the next. The GA moves from generation to generation until some specified stopping condition is met.

In building genetic algorithm, seven fundamental issues that affect the performance of the GA must be addressed: chromosome representation, initialization of the population, selection strategy, crossover and mutation, termination criteria, and evaluation measures.

This is an introduction to these issues of the GA.

a. chromosome representation:

Choosing an appropriate representation is the first step in applying the GA to any optimization problem (Grefenstte et al. 1985). Each individual or chromosome is made up of a sequence of genes from a certain alphabet. The alphabet can be a set of binary numbers, real numbers, integers, symbols (i.e., A, B, C, D), or matrices. For example, in a binary representation, individuals are represented by a collection of attributes or variables that are represented by a single binary string as follows:

Individual 001001

b. initialization:

Researchers use either heuristic or random techniques to generate feasible strings that form the initial population.

c. selection (reproduction):

Several types of reproduction (selection) methods are used to choose the individuals in the mating pool:

One way of selection is to use the roulette wheel sampling. This method assigns a pie-shaped slice of roulette to each member corresponding to its degree of fitness (see Figure 2.1), such that highly fit members get bigger slices than less fit ones. Next a random

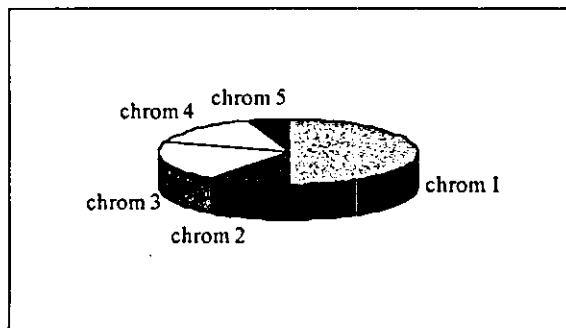


Figure 2.1: The roulette wheel

number is generated between 0 and the total fitness of the individuals. The winner is the one who's the roulette spinner ends up in. This method is good for giving higher probability of surviving to the fittest, but it has some additional computations and memory space needed.

In the ranking method, the roulette wheel selects two individuals with all members having the same weight, then the higher one in term of fitness is chosen to be included in the pool of alteration. This method may need less computations and memory space but still has a disadvantages of equaling the opportunities between highly fit and less fit ones is noticeable. Another way of selection is the Tournament Sampling. This method's principle is to choose a set of sequential individuals in a random place, and then the number with the highest fitness function is added to the mating list. This may be the simplest method of the selection methods that emphasize on the fitness in a simple manner.

d. crossover operator:

The widely crossover techniques used are:

1. Single point crossover: two selected strings are split at the same position, the front ends of both strings are recombined with the back ends of the other to generate two children of the next generation.

```

Parent 1: 0 1 1 0 0 1
Parent 2: 0 1 0 0 1 1
Child 1 : 0 1 1 0 1 1
Child 2 : 0 1 0 0 0 1

```

2. Uniform crossover: A mask is used to determine which parent propagates the bit at a particular position on its string to the same position on the string of one of the children, the other child inherits the bit of the other parent.

Parent 1: 0 1 1 0 0 1
Parent 2: 0 1 0 0 1 1
Mask : 0 0 1 1 0 0
Child 1: 0 1 0 0 0 1
Child 2: 0 1 1 0 1 1

e. mutation operator:

The widely mutation techniques used are:

1. The uniform mutation operator randomly selects one of the variables, v_i , from a parent and sets it equal to a random number uniformly distributed between the variable's lower and upper bounds.
2. The boundary mutation operator randomly selects one of the variables from a parent and randomly sets it equal to its lower or upper bound.

f. evaluation function:

A value for fitness is assigned to each solution (chromosome) depending on how it is close to solving the problem. These "solutions" are not to be confused with "answers" to the problem, think of them as possible characteristic that are employed in order to reach the answer.

g. termination:

The termination criteria may be a pre-specified maximum number of generations, or hybrid termination criteria where a target standard deviation value among the evaluation measures can be established on the basis of some arbitrarily 'acceptable' threshold. So, if the target value is reached stop, else continue till the maximum number of generations.

2.4 Literature Survey and Future Trends

2.4.1 Coarse-grained classification of grouping methods

There is no standard way of classifying cell-formation methods. A coarse-grained classification would result in the following three categories:

2.4.1.1 Visual Inspection

Visual Inspection or simply 'eye-balling' relies on the visual identification of machine cells and part families. Considerable experience is required in the process of identification even in small problem cases. However, as the size of the problem increases the task becomes almost impossible. Design drawings, process planning sheets or actual parts are used to identify a family of parts and the corresponding groups of machines.

2.4.1.2 Coding and Classification

In Coding and Classification approach, parts are coded by their manufacturing features, so parts having a similar feature code possess similar numbers, letters, etc. The three basic part features that can be classified and coded are:

1. shape or geometry of part,
2. function of part, and
3. manufacturing operations and tooling.

However, the information that can be incorporated into the part codes is not limited to these basic part features, e.g. production planning data can be added to the code systems. Unfortunately, genetic classification and coding system are hard to develop and expensive to maintain.

2.4.1.3 Production Flow Analysis

Production Flow Analysis (PFA) is a method of part-machine grouping which was developed by Burbidge in 1963 (Cesar and Ramachandran, 1994). PFA is particularly appealing since it does not require a complex part-coding system, it is relatively simple to implement, and can be applied to the organization of the existing production systems as well as to the design of new production systems. The method involves the following stages:

1. In the first stage, machines are classified by numbers according to type of operation that can be performed on them. Machines which can perform similar operations are usually classified with the same type number.
2. In the second stage, the parts list and the information in the production route card are carefully checked to identify and ensure correctness of the essential information for analysis. The essential information for each part consist of the operation to be performed on it and the machines necessary to perform each operation.
3. The third stage, factory flow analysis (FFA), involves macro examination of the part flow through the machines, this allows the problem to be decomposed into major machine-part groups. These three stages are merely the necessary preliminaries to provide the required data for the ultimate purpose of the analysis to determine appropriate machine-part subgrouping for cellular layout.
4. The last stage, involving the problem of identification of machine cell-part family, is the most difficult part of PFA. The difficulty comes from the amount of information that must be processed accurately to facilitate the formation of manufacturing cells that are dedicated to specific part-family production. The Genetic algorithm methodology presented in this thesis belongs to the family of production-based methods.

2.4.2 Fine-grained classification of grouping methods

A fine-grained classification of production-based methodologies for the solution of binary cell formation problems results in the following categories:

2.4.2.1 Array Based methods

The Array Based clustering methods perform a series of column and row manipulations to form part families and machine cells simultaneously. The Bond Energy Algorithm and the Rank Order Clustering Algorithm are two of the oldest and commonly array-based methods. In general Array Based techniques operate only on the part/machine incidence matrix and incorporate no other manufacturing information, such as limits on the size of each cell, machining times, etc. These methods require visual inspection of the solution to determine part families and machine cells and are affected by exceptional elements and bottleneck machines.

2.4.2.2 Hierarchical Clustering methods

Hierarchical Clustering methods can be divided into two sub-categories: divisive and agglomerative. Divisive algorithms start with all the machines (or parts) in a single group and then iteratively generate partitions until each machine is in a singleton cluster. Agglomerative algorithms start with each machine as an initial cluster and then proceed to merge groups into successively larger partitions until a single cluster contains the whole set. The similarity index used as the bases in the formation of groups by hierarchical clustering methods offers distinct advantage over array-based methods. It allows information, such as annual part demands, to be included in the clustering logic along with the routing sequences. These methods present the designer with a hierarchy of solutions from which the best alternative is chosen. However, these methods do not form the cells and families simultaneously, so additional methods must be employed to complete the design of the system. A dendrogram is typically used to represent the

solution hierarchy, making the choice of the correct solution difficult for large problems.

2.4.2.3 Graph Based approaches

Graph Based methods employ a graph or network representation of the cell formation problem and use partitioning techniques to create manufacturing cell. Mukhopadhyay et al. (2000) proposed an algorithm based on the Modified Hamiltonian Chain (MHC) and consists of two stages. Stage I forms the graph from the machine part incidence matrix. Stage II generates a Modified Hamiltonian Chain which is a subgraph of the main graph developed in Stage I, and it gives both the machine sequence and part sequence directly. In this proposed MHC method, the number of parts the machine pair 'i' and 'j' requires is used as the measure of strength. Dummy edges are considered in MHC for better accessibility in order to arrive at a block diagonal solution to the problem. This is a presentation of an approach by designing a MHC in the graph theoretic method to solve the group technology configuration problem.

2.4.2.4 Mathematical Programming methods

Clustering techniques are optimization and mathematical programming methods which offer several distinct advantages. These methods have the capability of constraining the size of each cell, the number of cells, as well as the ability to incorporate the use of different process plans for each of the parts. These formulations suffer from three critical limitations. First, because of the resulting nonlinear form of the objective function, most approaches do not concurrently group machines into cells and parts into families. Secondly, the number of machine cells must be specified in advance, which affects the grouping process and potentially obscures natural cell formation in the data. Thirdly

because the variables are constrained to integer values, most of these methods are computationally complex for large problems.

Mukattash (1997) derived a modified approach for solving the p-median model developed by Kusiak (1990), one of the most popular models which minimizes the total sum of distances between each pair of parts (machines). He developed a general algorithm with bounded and unbounded cell size, and based on this algorithm he developed heuristic algorithms to incorporate alternative process planes.

2.4.2.5 Artificial Intelligence techniques

Artificial Intelligence techniques were found to solve linear or non-linear problems. A very complicated non-linear problem can be easily solved or converged to the optimal or near-optimal solution in a very short time compared to other techniques, which are limited to a certain level of complexity and non-linearity. Artificial Intelligence (AI) techniques such as neural networks, fuzzy logic, and genetic algorithms, were used and tested in cell formation area. All of them have achieved satisfactory results.

542413

A new structure of the Hopfield neural network, Ortho-Synapse Hopfield Network (OSHN), was designed by Zolfaghari and Liangbeen (1997), for solving machine grouping problems. The OSHN, in conjunction with an objective-guided search scheme, has been implemented in an algorithm. The main advantages of the proposed approach lies in that it does not require the training process and can effectively handle the bottleneck machine problem. The proposed approach is computationally efficient, insensitive to the initial network state, and able to escape local optimum solutions through the global optimum solution may not be guaranteed.

Malave and Ramachandran (1994) used neural networks based on a simple learning algorithm to form machine/part cells/families. From the part machine incidence matrix, a set of binary vectors can be identified in which each vector represents a particular part. These vectors are called characteristic vectors, since they characterize parts. This set of vectors form the training set for the neural network. At steady state, each of these characteristic vectors is associated with a particular output layer neuron. The group of vectors which are associated with a particular output layer neuron form a part family. Machine cells are formed by assigning the machines among the various part group formed. The assignment of machine cells can be done by observing the steady state weight matrix, because the steady state weights of the neural network contain information on how the machines are distributed among the various part group formed.

Joines et al. (1996) presented an integer program that is solved using a genetic algorithm to assist in the design of cellular manufacturing systems. The objective function was to minimize intercellular movements of parts. He used a combination of six crossover operators and four mutation operators. His approach offers improved design flexibility by allowing a variety of evaluation functions to be employed and by incorporating design constraints during cell formation.

Zhao et al. (1995) presented a genetic algorithm for fuzzy clustering. Fuzzy clustering is one approach for a more accurate presentation of clustering problems based on uncertain or inexact real-data structures. The ordinary crisp clustering methods restrict that each point of the data set belongs to exactly one cluster, however there may exist points whose lineage are much less evident, here, the fuzzy set theory provides a means for a more accurate presentation of uncertain or inexact information. The fuzzy rules are learned and used to control various parameters based on the relative performance of the

best, worst and mean of the current population such as mutation rate which is altered continuously in order to explore more and more hyperplanes.

Cheng et al. (1998) suggested that the cellular manufacturing formation problem may be formulated as a Travelling Salesman Problem (TSP). So, the problem will be solved if the associated TSP is solved. Cities in TSP corresponds to machines or parts. A solution methodology based on genetic algorithms is proposed to solve the TSP-cell formation problem. The objective function was to minimize intercellular movements of parts. He used path representation, edge recombination crossover operator, no mutation, fitness ranking reproduction and elitism replacement approach to solve the GA.

Dimopoulos and Mort (2001) proposed a genetic programming method which is similar to any other evolutionary computation method in the sense that the Darwinian principle of strife for survival is employed during the search for an optimal solution. However, while most evolutionary algorithms evolve fixed-length strings of binary, integer or real numbers, genetic programming evolves solutions in the form of computer programs of variable length. They used a Hierarchical Clustering methodology based on genetic programming for the solution of simple cell-formation problems. Their methodology comprised of two main stages. First, a similarity coefficient was calculated for each pair of machines that were available in the plant. The value of the coefficient represented the similarity of the machines in terms of the common operations performed. Then a dendrogram that linked individual machines or group of machines according to the values of their similarity coefficients is generated. The main disadvantage is its computational complexity in relation to other cell-formation methodologies.

In this thesis, the genetic algorithm methodology is used to find near-optimal solutions to the cell formation problem using the roulette wheel reproduction approach, a combination of crossover techniques, a combination of mutation techniques, elitism replacement and hybrid termination strategy. In addition it uses two objective functions. The first is grouping efficacy, which attempts to minimize the number of voids plus the number of exceptional elements divided by the total operational zone. The operational zone is defined by the number of operations (exceptional elements plus all ones along the diagonals). The other is the Travelling Salesman Problem formulation which attempts to determine the desirable permutation for rows and columns in a solution matrix by using a distance (similarity) measures between a pair of rows (machines) and columns (parts). A metalevel genetic algorithm is used to optimize the control parameters i.e. population size, target standard deviation, crossover rate and mutation rate.

2.5 Future Trends

The future trends are going toward constructing hybrid artificial intelligence systems, which include two AI techniques. For example, neural networks learn using genetic algorithms. This means that the weights and biases of all the neurons are joined to create a single vector. A certain set of vectors is a correct solution to the problem at hand. So, the evolutionary algorithm is used to find one of these vectors. It may be necessary to develop a “hybrid” system, which combines several types of intelligence to solve the cell formation problem.

Another active area in the cell formation problem is to include multi-objective evaluation function, such as process plans, scheduling constraints, volume of production, sequencing, etc. The inclusion of these additional factors in the design of part families

and machine cells is a more realistic approach toward achieving the future benefits of group technology.

Applying metalevel GA to optimize the genetic algorithm operators i.e. crossover techniques combination, mutation techniques combination etc. may increase the possibility to find the optimal solution. Also metalevel GA can be used to optimize the GA parameters i.e. generation gap (the generation gap controls the percentage of the population to be replaced during each generation), population size, mutation rate, crossover rate, etc. This is essential since better solutions to the cell formation problem using GA are required.

2.6 Summary

This chapter provided definition and description of the cell formation problem. In addition, an introduction to genetic algorithms, GA definition, advantages, concepts, methodology, and mathematical foundation are presented. Also, it provided a literature survey of the previous work in the manufacturing cell formation problem solution methodologies and future trends in this field.

In the next chapter, the problem of manufacturing cell formation will be formulated to a genetic algorithm.

3- CELL FORMATION

PROBLEM FORMULATION

3.1 Introduction

The formulation and implementation of GA to solve the cell formation problem required building a MATLAB software, which includes several graphical user interfaces (GUI's) and several functions. To maximize the cell formation flexibility, an implementation was adopted that allows the user to employ various representation schemes, sets of operators, and evaluation measures. This flexibility is gained through the implementation of the Genetic Algorithms software using MATLAB.

The GA software was tested on ten data sets from the literature to study its effectiveness as a clustering tool. In all experiments the GA employed both GEC (grouping efficacy) and TSP (Travelling Salesman Problem-cell formulation) objective functions and their initialization, mutation and crossover techniques.

The core of the problem formulation lies in building a genetic algorithm function, which should perform the GA procedures utilizing the data sets from the literature to generate an initial population of solutions (chromosomes) randomly. These solutions are then reproduced in order to find the optimal or near optimal solution. A separate function is constructed to handle the outputs and to display the results including grouping efficacy, grouping efficiency, TSP results, number of voids, number of exceptional elements, number of cells and the number of iterations required to find near optimal solution. Also it display a plot of the maximum, the average and the standard deviation of the fitness values variation with the number of generations.

The flow diagram shown in Figure 3.1 represents the main menu of the GA software. The details about the computer implementation of the genetic algorithm to solve the cell formation problem using MATLAB is presented in appendix A.

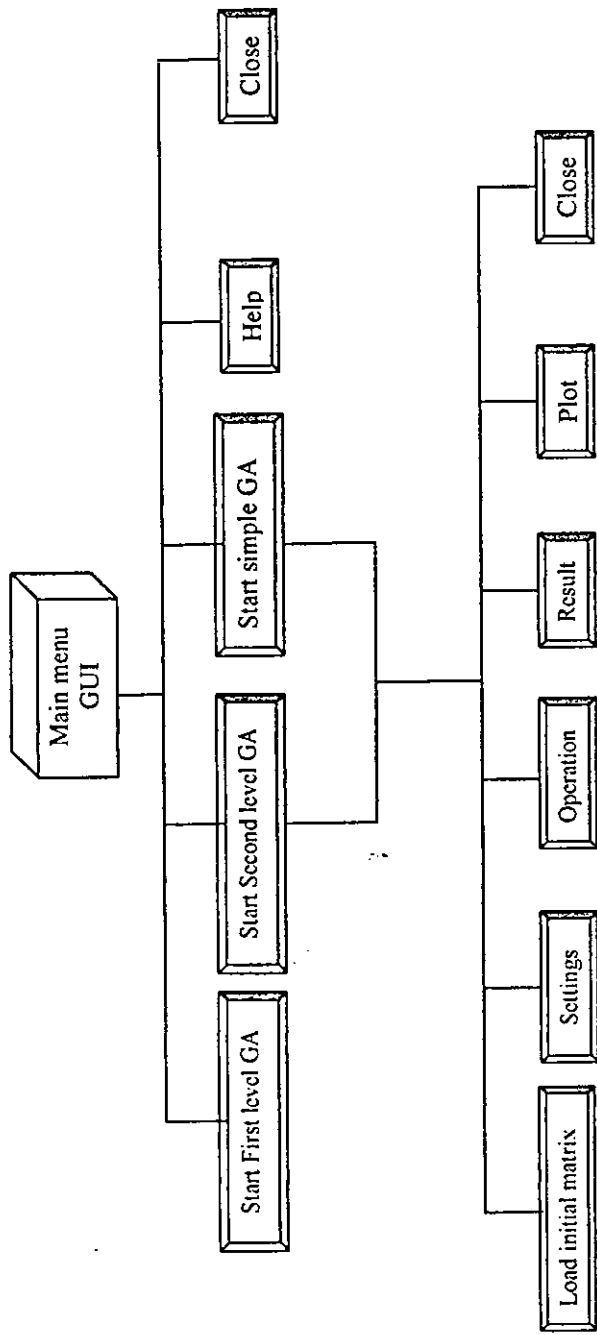


Figure 3.1: The main menu of MATLAB software

The main system functions, which are constructed to handle, analyze and find the optimal or near optimal solutions are the following (see Figure 3.2):

1. The initial matrix (data set) functions
2. Genetic algorithm functions
3. Experimental results (GUI)

In the following sections these functions are discussed and explained.

3.2 Initial Matrix (Data Set) Functions

Finding a set of problems for the evaluation of an optimization method is always a difficult task. The main requirements that a representative set of test problems should fulfill are the following:

- Different instances of the problem should be included in terms of size, difficulty or any other parameter that can be varied.
- Results from alternative solution methods should be available, so that meaningful comparisons can be made.

In the case of cell-formation problems, there is no formal definition of the difficulty of a particular instance of the problem. The lack of parameter-based estimation of the difficulty of the problem means that the creation of randomly generated problems is not as straightforward as in the case of other optimization problems. In practice, researchers of the cell-formation problem evaluate their methods on test problems taken from the literature, so there are many comparative results available. While the second requirement for an appropriate set of test problems is fulfilled, there should be a careful qualitative consideration of the cases that will be chosen in order to be as close as possible to the fulfillment of the first requirement as well. The data sets used to evaluate this GA clustering technique are of varying size. In each case, correct grouping is defined as

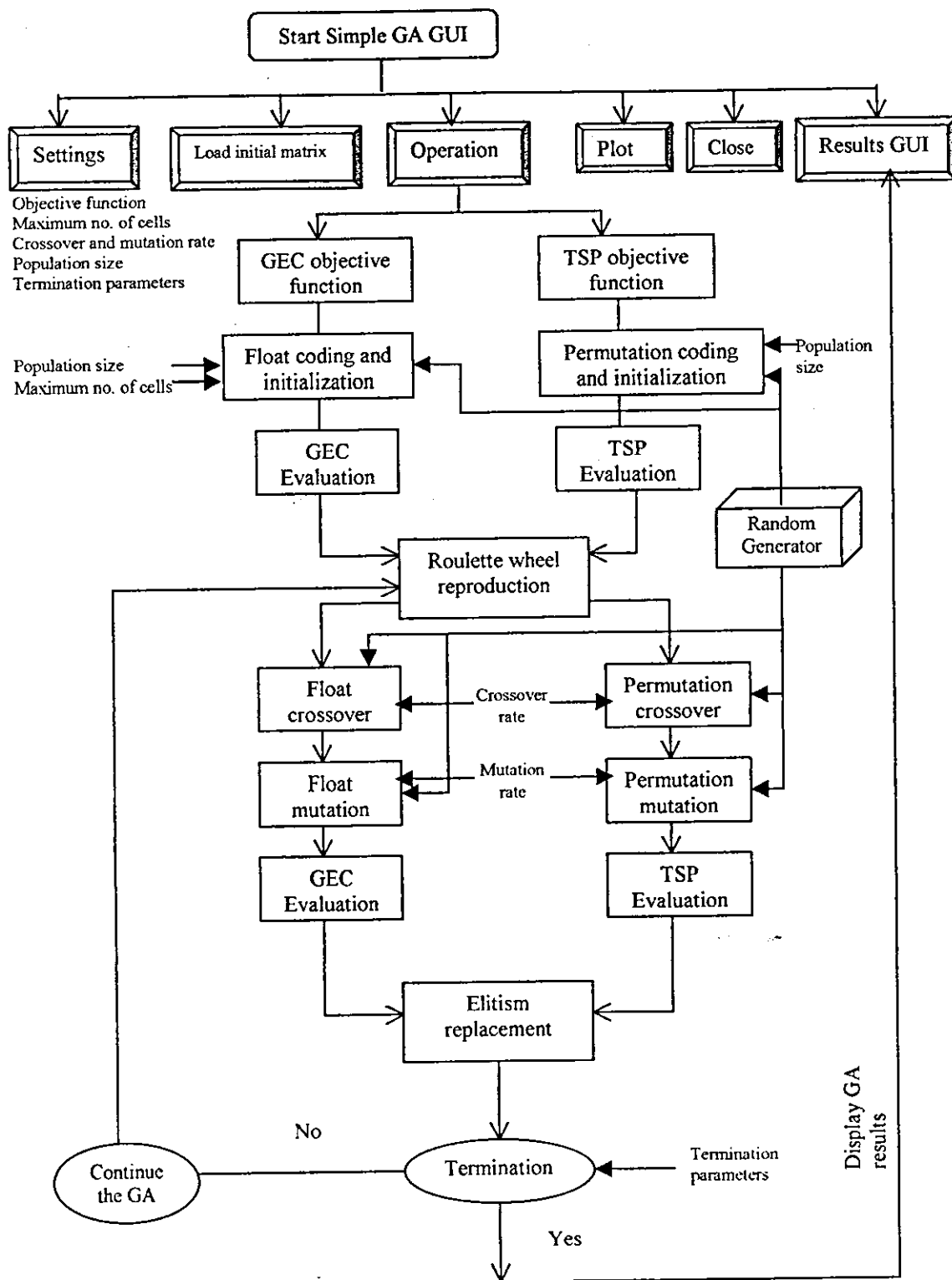


Figure 3.2: The simple GA flow diagram

being equivalent to the best known configuration provided in the literature, on the basis of grouping efficacy, grouping efficiency and the cell index.

The inputs to the above algorithms come from company's technological experience and knowledge specified in route-sheets. This knowledge is presented in the literature in the form of a zero-one matrix. Companies usually forbid the release of their products route-sheet because of competition unless it is an out of date information. This reason motivated researchers to design and publish theoretical problems similar to real ones in the form of zero-one matrix in order to study the performance of their algorithms compared to others.

Ten notable problems have been selected from the literature (see Table 3.1) to test the performance of the algorithm. Alternative solutions can be obtained by using pre-determined maximum number of cells or leaving the final number of cells unrestricted.

Table 3.1: Benchmark test problems

No	Incidence matrix	Reference	Size
1	King (1980a)	Mukattash (1997)	6x10
2	Chen and Guerrero (1994)	Mukattash (1997)	6x15
3	Chu and Hayya (1991)	Zhao et al. (1995)	9x9
4	King (1980b)	Mukhopadhyay et al. (2000)	10x8
5	Viswanathan (1996)	Mukattash (1997)	10x12
6	Simple Chan and Milner (1982)	Malave and Ramachandran, (1994)	15x10
7	Complex Chan and Milner (1982)	Malave and Ramachandran, (1994)	15x10
8	Chandrasekharan and Rajagopalan (1986)	Cheng et al. (1998)	8x20
9	King (1980c)	Malave and Ramachandran, (1994)	14x24
10	Burbidge (1960)	Jeffrey et al. (1996)	20x35

3.3 Genetic Algorithm Functions

In order to provide a comprehensive understanding of our algorithm a simple illustrative example is used throughout. Table 3.2 shows the initial matrix of machines (rows) and the corresponding parts (columns) processed by these machines.

Specifically, for the cell formation problem, an integer alphabet $\{1, \dots, k\}$ is employed, where k represents an upper bound on the number of part families/machine cells. For an individual, the first m variables represent the machines, and the last n variables are associated with the parts. Therefore, each individual is a vector of $m + n$ integer variables.

$$\text{Chromosome1} = x_1 x_2 \dots x_m, y_1 y_2 \dots y_n$$

In our illustrative example $m=3$, $n=5$ and $k=2$

$$\text{Chromosome1} = 1 1 2, 2 1 1 1 2$$

Each part and machine variable is equal to the number of its assigned family or cell, see Table 3.3. In the above example, $y_2=1$ indicates that part 2 is assigned to part family 1, while $x_3=2$ indicates that machine 3 is assigned to machine cell 2. Part families are assigned to the machine cell with the same number.

Chromosome no.	Initial population (randomly selected)
Chromosome1	1 2 1, 2 2 1 1 2
Chromosome2	2 2 1, 2 2 2 2 1
Chromosome3	1 2 2, 1 2 1 2 1
Chromosome4	1 1 2, 2 1 2 2 2
Chromosome5	1 1 1, 1 2 2 1 1
Chromosome6	2 1 1, 2 2 1 2 1
Chromosome7	2 2 1, 2 1 1 1 1
Chromosome8	1 2 2, 1 1 1 1 2
Chromosome9	2 1 2, 2 1 2 2 2
Chromosome10	2 1 2, 1 2 1 2 1

However, the selection of the representation of the problem requires careful consideration. Without an appropriate representation, genetic search can be slow and produce mediocre results. With a good representation and appropriate operators, convergence to high quality solutions can be fast in highly complex and constrained problems. Although several genetic representations have been used for the Travelling

Salesman Problem the permutation representation is both common and simple. In this representation the chromosome is stored as a list (path) of ordered machines or parts.

Each gene in a chromosome may correspond to a machine or a part.

For example, chromosome1 represents a path of machines, while chromosome2 represents a path of parts. We will take parts for illustration and the same is applicable for machines.

chromosome1 : 3 1 2

chromosome2 : 3 5 4 2 1

The initialization function produces randomly a number of individuals equal to the population size specified by the user, which will be ten for our example. Tables 3.3 and 3.4 show the initial populations of size 10 for both GEC and TSP objective functions.

Chromosome no.	Initial population (randomly selected)
Chromosome1	5 4 3 1 2
Chromosome2	3 2 1 4 5
Chromosome3	5 4 1 2 3
Chromosome4	4 5 1 2 3
Chromosome5	5 1 2 3 4
Chromosome6	1 2 4 5 3
Chromosome7	5 3 2 4 1
Chromosome8	3 5 2 1 4
Chromosome9	4 1 2 3 5
Chromosome10	2 4 3 5 1

3.3.2 Evaluation function

Evaluation functions (fitness function) of many forms can be used in a GA, subject to the minimal requirement that the function can map the population into a partly ordered set.

Cell formation in a given machine-part incidence matrix involves rearrangement of rows and columns of the matrix to create machine cells and part families (i.e. blocks). The major objective is to reduce intercellular moves among machine cells. In a solution matrix, a block diagonal form (BDF) is often desirable because the blocks may be easily identified to facilitate the subsequent cell formation decisions. The two objective functions used in this thesis are discussed below.

3.3.2.1 Grouping efficacy

Grouping efficacy is chosen as the initial evaluation measure because it has been used frequently in the literature and results are available for comparison. It seeks to minimize the number of exceptional elements and the number of voids (zeros) in the diagonal blocks. Exceptional elements present inter-cell movements of parts, which reduce the effectiveness of cellular manufacturing. More specifically, grouping efficacy attempts to minimize the number of voids plus the number of exceptional elements divided by the total operational zone. The operational zone is defined by the number of operations (exceptional elements plus all ones along the diagonal blocks). Grouping efficacy has a value of one when there are no exceptional elements and no voids and a value of zero if the number of exceptional elements equals the total number of operations. Formally, grouping efficacy (Γ) is defined as:

$$\Gamma = \frac{1 - \phi}{1 - \phi} \dots\dots\dots(3.1)$$

$$\phi = \frac{e_o}{e} \dots\dots\dots(3.2)$$

$$\phi = \frac{e_v}{e} \dots\dots\dots(3.3)$$

where,

e_o : number of 1's outside the machine/part groups (exceptional elements).

e_v : number of 0's within the machine/part groups (voids).

e : total number of operations (ones) in the incidence matrix.

Γ : grouping efficacy.

Where φ is the ratio of the number of exceptional elements to the total number of operations and ϕ is the ratio of the number of voids in the diagonal blocks to the total number of operations. This expression has the requisite properties like non-negativity and zero to one range. Moreover, φ and ϕ are the only ratios and are not affected by the size of the matrix.

Properties of the grouping efficacy function:

The grouping efficacy function is basically a ratio of two independent ratios that are dependent upon the block-diagonalized matrix.

1. The value of GEC varies ranging from '0' to '1'. By definition $0 \leq \varphi \leq 1$ and $0 \leq \phi \leq 1$ therefore $0 \leq \Gamma \leq 1$.
2. From this function, physical significance can be obtained from its different extreme values. When $\Gamma=0$, i.e. zero efficacy, $\varphi=1$ which implies that all the '1's in the matrix are outside the diagonal blocks. On the other hand $\Gamma=1$ implies that $\phi = 0$ meaning a perfect block-diagonalization which can be further interpreted as- there are no 1's outside the diagonal blocks meaning no existence of a bottleneck problem in the system.

Fitness function is used to evaluate the value of the individuals within the population. According to the fitness value scored, individuals are selected as parents to produce offsprings in the next generation or to disappear in the next generation.

The calculation of the GEC values associated with each chromosome in Table 3.3 is illustrated here:

Table 3.5 shows the intermediate solution for chromosome2: 2 2 1, 2 2 2 2 1. Note that the first cell contains machine 3 and part 5, and cell two contains all the other machines and parts. The values of $e_0=2$, $e_v=3$ and $e=8$, then $\Gamma=(1-2/8)/(1+3/8)=0.546$. Table 3.6 gives all the GEC values associated with the ten chromosomes.

Table 3.5: Intermediate solution

	p5	p1	p2	p3	p4
m3	1	0	1	0	0
m1	1	1	1	1	0
m2	0	1	0	0	1

Table 3.6: Evaluation of GEC objective function.

No.	Initial population	f(x) from eq.(3.1)
1	1 2 1, 2 2 1 1 2	0.154
2	2 2 1, 2 2 2 2 1	0.546
3	1 2 2, 1 2 1 2 1	0.500
4	1 1 2, 2 1 2 2 2	0.167
5	1 1 1, 1 2 2 1 1	0.357
6	2 1 1, 2 2 1 2 1	0.250
7	2 2 1, 2 1 1 1 1	0.400
8	1 2 2, 1 1 1 1 2	0.400
9	2 1 2, 2 1 2 2 2	0.308
10	2 1 2, 1 2 1 2 1	0.250

3.3.2.2 Traveling Salesman Problem:

To increase the flexibility of the MATLAB software an alternative objective function is used which is the TSP formulation, where cities in the TSP corresponds to machines and parts. In the TSP, the total distance is calculated as the distance traveled from the starting city to the last city plus the distance from the last city back to the starting city. In the

TSP-cell formation problem, returning to the starting city (i.e. machine or part) does not have any practical meaning. As the first and last machines/parts need not be connected, the total distance of a path instead of a tour is calculated in our fitness function.

The problem of arranging rows and columns is similar to a permutation problem. To determine the desirable permutation for rows and columns in a solution matrix, a distance (similarity) measure between a pair of rows (machines) i and j is defined as follows:

$$d_{ij} = \sum_{k=1}^n |a_{ik} - a_{jk}| \quad \dots\dots\dots(3.4)$$

Similarly, we use the following distance measure for a pair of parts i and j :

$$d'_{ij} = \sum_{k=1}^m |a_{ki} - a_{kj}| \quad \dots\dots\dots(3.5)$$

In this proposed TSP, we used the number of parts (machines) the machines (parts) pair ‘ i ’ and ‘ j ’ requires as a measure of strength. 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 machine cells. This will increase material handling. Therefore, a manufacturing cell formation algorithm must place machines processing similar parts (and parts requiring similar machines) close to one another in the final permutation. This in turn attempts to minimize the total distance between pairs of machines.

Let ρ be the permutations of machines and σ be the permutations of parts. For a permutation of machines: 2 3 1, $\rho(1)$ is 2, and $\rho(3)$ is 1. The proposed approach converts

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

$$\text{Total distance} = \sum_{i=1}^{m-1} \sum_{k=1}^n |a_{\rho(i)\sigma(k)} - a_{\rho(i+1)\sigma(k)}| \dots\dots\dots(3.6)$$

Where,

m: number of machines.

n: number of parts.

$\rho(i)$: the permutation of machines.

$\sigma(k)$: the permutation of parts.

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

By minimizing the total distance, machines that process similar parts are grouped together. To diagonalize a matrix, we may rearrange the parts using the proposed approach:

$$\text{Total distance} = \sum_{k=1}^{n-1} \sum_{i=1}^m |a_{\rho(i)\sigma(k)} - a_{\rho(i)\sigma(k+1)}| \dots\dots\dots(3.7)$$

The greater the number of parts required by that machine pair, the stronger the pair becomes. This method is considered to be a simple one and proved to be quite efficient in arriving at a solution as block diagonal form (BDF). The machine part incidence matrix only reveals the machines required for each part, it does not reveal any sequence through which the part has to move. The proposed algorithm also gives the solution in the BDF, considering the similarity of the machine required, without considering other costs involved.

The TSP will give the desired machine sequence and part sequence directly. If the machine part incidence matrix is drawn with the generated sequences, it gives the solution in BDF. Weak edges are the edges where partitioning into cells takes place, i.e. the distance measure associated with that partition is a maximum. The strength of this algorithm is that we do not need to identify the edges where to partition into cells. If all the machines and parts are arranged as appears in the sequences, cells and part families will automatically show in BDF.

The first step to find the TSP values associated with each chromosome in Table 3.4 is to calculate the distance matrix for both machines and parts based on equations 3.4 and 3.5. The distance matrix for the parts is shown in Table 3.7. The fitness value of chromosome 8 : 3 5 2 1 4 is calculated as follows :

$$\text{Total distance} = 1 + 0 + 2 + 1 = 4.$$

Table 3.7: Distance matrix for parts.

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

The weak edge in this sum is the maximum distance measure i.e. 2. This will partition the permutation to two families. Family one contains parts 5, 3, 2 and the other contains 1 and 4. If the machines sequence is calculated to be 3 1 2, and the matrix is rearranged in these two sequences the intermediate solution will be as in Table 3.8, note that this is the optimal solution. Table 3.9 shows the TSP values for the ten chromosomes.

Table 3.8: Intermediate solution

	p3	p5	p2	p1	p4
m3	1	0	1	0	0
m1	1	1	1	1	0
m2	0	0	0	1	1

Table 3.9: Evaluation of TSP objective function.			
No.	Initial population	f(x) value from eq.(3.7)	g(x)= 1/f(x)
1	5 4 3 1 2	8	0.125
2	3 2 1 4 5	7	0.143
3	5 4 1 2 3	7	0.143
4	4 5 1 2 3	8	0.125
5	5 1 2 3 4	7	0.143
6	1 2 4 5 3	9	0.111
7	5 3 2 4 1	6	0.167
8	3 5 2 1 4	4	0.250
9	4 1 2 3 5	5	0.200
10	2 4 3 5 1	8	0.125

3.3.3 Reproduction (Parent selection technique)

Reproduction is a process in which individual strings are copied according to their objective function values, i.e., fitness values. Copying strings according to their fitness values means that strings with a higher value have a higher probability of contributing one or more offspring in the next generation. This operator is an artificial version of natural selection. The reproduction operator receives the matrix of the generated solutions. Each solution should enter the objective functions GEC or TSP, which are considered as the fitness function for this GA.

The roulette wheel method of reproduction is to be used in this research. It depends upon the number of occurrences of each solution in a certain generation. To compute this, a probability of selection for each solution should be calculated from the fitness value of that solution as shown below. Also, the expected count of the solution found from the fitness value and the average fitness should be calculated in order to find the actual count of the solution which determines the number of copies for each solution in the next generation.

The probability of selection should be calculated for each solution i as follows:

$$P_{select_i} = \frac{f_i}{\sum f_i} \dots\dots\dots(3.8)$$

Where,

f_i : fitness value of solution i

$\sum f_i$: total fitness for all solutions (chromosomes).

The expected count is then found for each solution i to determine its actual occurrences in the generation as follows:

$$E(\text{count } i) = \frac{f}{\bar{f}} \dots\dots\dots (3.9)$$

where,

\bar{f} = average of the selection probabilities.

Equation (3.9) should be applied to each solution as shown in Tables 3.10 and 3.11. The expected count value will determine the number of occurrences of each solution in the mating pool. According to the expected count value of each solution, a mating pool is to be constructed to do the crossover and mutation. Each solution should be repeated $\{E(\text{Count})\}$ times to form the reproduced generation. A generation is defined as a set of cell formation solutions determined by the initial population size of the genetic algorithm, this set of solutions contain the genetic materials that will be handled by the genetic algorithm through all GA iterations.

3.3.4 Crossover and mutation

The purpose of crossover is to create children whose genetic material resembles their parent's genes in some fashion. Crossover combines building blocks of good solutions from different chromosomes. Crossover is not used to produce all members of the

Table 3.10: The roulette wheel reproduction for the GEC objective function.

Chromosome no.	Initial population (randomly selected)	f(x) from equation (3.1)	Pselect $f/\sum f$	Expected count $f(x)/f$	Actual count	Mating pool after reproduction
1	1 2 1, 2 2 1 1 2	0.154	0.0462	0.462	0	2 2 1, 2 2 2 2 1
2	2 2 1, 2 2 2 2 1	0.546	0.1638	1.638	2	2 2 1, 2 2 2 2 1
3	1 2 2, 1 2 1 2 1	0.500	0.1501	1.501	2	1 2 2, 1 2 1 2 1
4	1 1 2, 2 1 2 2 2	0.167	0.0500	0.500	0	1 2 2, 1 2 1 2 1
5	1 1 1, 1 2 2 1 1	0.357	0.1072	1.072	1	1 1 1, 1 2 2 1 1
6	2 1 1, 2 2 1 2 1	0.250	0.0751	0.751	1	2 1 1, 2 2 1 2 1
7	2 2 1, 2 1 1 1 1	0.400	0.1201	1.201	1	2 2 1, 2 1 1 1 1
8	1 2 2, 1 1 1 1 2	0.400	0.1201	1.201	1	1 2 2, 1 1 1 1 2
9	2 1 2, 2 1 2 2 2	0.308	0.0924	0.924	1	2 1 2, 2 1 2 2 2
10	2 1 2, 1 2 1 2 1	0.250	0.0751	0.751	1	2 1 2, 1 2 1 2 1
Sum		3.3308	1.000	10.00	10	
Average		0.3331	0.100	1.000	1.0	
Max		0.546	0.163	1.638	2.0	

Table 3.11: The roulette wheel reproduction for the TSP objective function.

Chromosome no.	Initial population (randomly selected)	f(x) from equation (3.7)	$g(x) = 1/f(x)$	Pselect $g(x)/\sum g$	Expected count $g(x)/g$	Actual count	Mating pool after reproduction
1	5 4 3 1 2	8	0.125	0.0816	0.816	1	5 4 3 1 2
2	3 2 1 4 5	7	0.143	0.0933	0.933	1	3 2 1 4 5
3	5 4 1 2 3	7	0.143	0.0933	0.933	1	5 4 1 2 3
4	4 5 1 2 3	8	0.125	0.0816	0.816	1	4 5 1 2 3
5	5 1 2 3 4	7	0.143	0.0933	0.933	1	5 1 2 3 4
6	1 2 4 5 3	9	0.111	0.0726	0.726	0	3 5 2 1 4
7	5 3 2 4 1	6	0.167	0.1088	1.088	1	5 3 2 4 1
8	3 5 2 1 4	4	0.250	0.1633	1.633	2	3 5 2 1 4
9	4 1 2 3 5	5	0.200	0.1306	1.306	1	4 1 2 3 5
10	2 4 3 5 1	8	0.125	0.0816	0.816	1	2 4 3 5 1
Sum		1.5313	1.000	1000	10		
Average		0.1531	0.100	1.000	1.0		
Max		0.250	0.163	1.633	2.0		

next generation. The crossover rate is the proportion of the next generation produced by crossover, which will be 0.6 for our illustrative example. So, the number of chromosomes that will be crossed over = crossover rate x population size = 0.6 x 10 = 6

Mutation 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. The mutation manipulates in some members of the population and puts it in the new population to be evaluated. Normally, the mutation works on each member separately, not like the crossover, which works on pairs. Mutation is used to rejuvenate the search, extending the search into previously unexplored areas.

If mutation probability $P_m = 0.1$, then the number of bits to undergo mutation (NM) = $(P_m) * (\# \text{ of bits transferred})$. For the GEC objective function the number of bits transferred = (number of machines + number of parts) x population size = $8 \times 10 = 80$, so $NM = 0.1 \times 80 = 8$. But for the TSP $NM = 5 \times 10 \times 0.1 = 5$ bits will be mutated.

In this thesis, the GA utilizing the GEC uses one crossover and a combination of two mutation techniques, while that utilizes TSP uses combinations of three crossover and five mutation techniques. So, the number of chromosomes that will be crossed over or mutated is partitioned equally between these techniques.

The crossover and mutation techniques used are dependent on the representation of the problem. The float representation of the GEC objective function and the permutation representation of the TSP are discussed in details in section 3.3.1.

3.3.4.1 Float representation

The following crossover and mutation techniques are applied to the chromosomes in the mating pool as shown in Table 3.12.

The simple float crossover operator randomly selects a cut point. The two parents are then split at this point, and concatenating the segments from both parents creates children. For example chromosomes 1 and 2 (Table 3.12, column 5) are crossed over at site 4:

Chromosome 1: 2 1 1, 2 2 1 2 1

Chromosome 2: 2 2 1, 2 1 1 1 1

The following two offsprings are created from these two parents:

Offspring 1: 2 1 1, 2 1 1 1 1

Offspring 2: 2 2 1, 2 2 1 2 1

Table 3.12: Crossover and mutation in the GEC objective function.

No	Mating pool after reproduction	Mate (Random selection)	Cross-over site	Before crossover	After crossover	After mutation
1	2 2 1, 2 2 2 2 1	4	2-4	2 1 1, 2 2 1 2 1	2 1 1, 2 1 1 1 1	2 1 1, 2 1 1 2 1
2	2 2 1, 2 2 2 2 1	-	2-4	2 2 1, 2 1 1 1 1	2 2 1, 2 2 1 2 1	2 2 1, 2 2 1 2 1
3	1 2 2, 1 2 1 2 1	3	-	1 2 2, 1 2 1 2 1	1 2 2, 1 2 2 2 1	1 2 2, 1 2 2 1 1
4	1 2 2, 1 2 1 2 1	5	2-5	2 2 1, 2 2 2 2 1	2 2 1, 2 2 1 2 1	1 2 1, 2 1 1 2 1
5	1 1 1, 1 2 2 1 1	6	2-5	1 2 2, 1 2 1 2 1	1 1 1, 1 2 2 1 1	1 1 1, 1 2 2 1 1
6	2 1 1, 2 2 1 2 1	1	1	1 1 1, 1 2 2 1 1	1 2 2, 1 2 1 2 1	1 2 2, 1 2 1 2 2
7	2 2 1, 2 1 1 1 1	2	1	2 2 1, 2 2 2 2 1	2 2 1, 2 2 2 2 1	2 2 1, 2 2 2 2 1
8	1 2 2, 1 1 1 1 2	-	-	1 2 2, 1 1 1 1 2	1 2 2, 1 1 1 1 2	1 2 2, 1 1 1 1 2
9	2 1 2, 2 1 2 2 2	-	-	2 1 2, 2 1 2 2 2	2 1 2, 2 1 2 2 2	2 1 2, 2 1 2 1 2
10	2 1 2, 1 2 1 2 1	-	-	2 1 2, 1 2 1 2 1	2 1 2, 1 2 1 2 1	2 1 2, 1 2 2 2 1

The two mutation techniques used are:

- The uniform mutation operator randomly selects one of the variables, v_i , from a parent and sets it equal to a random number uniformly distributed between the variable's lower (a_i) and upper (b_i) bounds. Where $a_i=1$ and $b_i=2$, in our example.

Chromosome 1: 2 2 1, 2 2 1 2 1

New : 1 2 1, 2 1 1 2 1

- The boundary mutation operator randomly selects one of the variables from a parent and randomly sets it equal to its lower (a_i) or upper bound (b_i).

Chromosome 3: 1 2 2, 1 2 2 2 1

New : 1 2 2, 1 2 2 1 1

3.3.4.2 Order-based representation

The following crossover and mutation techniques are applied to the chromosomes in the mating pool as shown in Table 3.13.

1. Single point crossover:

After reproduction, single point crossover may proceed in two steps. First, members of the newly reproduced strings in the mating pool are mated at random. Second, each pair of strings undergoes crossing over as follows: an integer position k along the string is selected uniformly at random between 1 and the string length less one $[1, L-1]$. Till this point the permutation is copied from the first parent, then the second parent is scanned and if the number is not yet in the offspring it is added. For example the two chromosomes 1 and 2 in Table 3.13, column 5 are crossed at site 2 as follows:

Chromosome 1: 3 5 2 1 4

Chromosome 2: 5 3 2 4 1

Offspring 1 : 3 5 2 4 1

Offspring 2 : 5 3 2 1 4

Table 3.13: Crossover and mutation in the TSP objective function.

No	Mating pool after reproduction	Mate (Randomly selected)	Cross-over site	Before crossover	After crossover	Before mutation	After mutation
1	5 4 3 1 2	4	2-4	3 5 2 1 4	3 5 2 4 1	3 5 2 4 1	3 5 2 4 1
2	3 2 1 4 5	3	2-4	5 3 2 4 1	5 3 2 1 4	5 3 2 1 4	5 4 2 1 3
3	5 4 1 2 3	-	-	5 4 3 1 2	5 2 1 4 3	5 2 1 4 3	5 2 1 4 3
4	4 5 1 2 3	5	2-5	3 2 1 4 5	2 4 3 1 5	2 4 3 1 5	2 3 4 1 5
5	5 1 2 3 4	6	2-5	4 5 1 2 3	4 5 1 2 3	4 5 1 2 3	4 5 1 2 3
6	3 5 2 1 4	1	2	5 1 2 3 4	5 4 1 2 3	5 4 1 2 3	5 2 1 4 3
7	5 3 2 4 1	2	2	5 4 1 2 3	5 4 1 2 3	5 4 1 2 3	5 4 1 2 3
8	3 5 2 1 4	-	-	3 5 2 1 4	3 5 2 1 4	3 5 2 1 4	3 1 2 5 4
9	4 1 2 3 5	-	-	4 1 2 3 5	4 1 2 3 5	4 1 2 3 5	4 1 2 3 5
10	2 4 3 5 1	-	-	2 4 3 5 1	2 4 3 5 1	2 4 3 5 1	2 5 1 4 3

2. Partially-Mapped Crossover (PMX)

To exchange ordering and value information among different strings we present a new genetic operator called partially-mapped crossover, because a portion of one string ordering is mapped to a portion of another and the remaining information is exchanged after appropriate swapping operations (Goldberg and Lingle, 1985).

Chromosome 3: 5 4 3 1 2

Chromosome 4: 3 2 1 4 5

PMX proceeds as follows. First, two positions are chosen along the string uniformly at random. The substrings defined from the first number to the second number are called the *mapping sections*. Next, we consider each mapping section separately by mapping the other string to the mapping section through a sequence of swapping operations. For example, if we pick two random numbers say 2 and 4, this defines the two mapping sections, 4-3-1 in Chromosome 3 (in the above example), and 2-1-4 in Chromosome 4. The mapping operation, say from Chromosome 3 to Chromosome 4, is performed by swapping first the 4 and the 2, the 3 and the 1, and the 1 and the 4, resulting in a well-defined offspring. Similarly the mapping and swapping operation of Chromosome 4 to Chromosome 3 results in the swap of the 2 and the 4, the 1 and the 3, and the 4 and the 1. The resulting two new offsprings are as follows:

Offspring 1: 5 2 1 4 3

Offspring 2: 2 4 3 1 5

3. Order Based Crossover:

Given two chromosomes, a subset of permutations is chosen from the list. To create the first offspring the first chromosome is copied and the selected subset of parts/machines

is reordered to match that found in the second chromosome. The second offspring is similar, a copy of the second chromosome with the first chromosome ordering imposed.

Chromosome 5: 4 5 1 2 3

Chromosome 6: 5 1 2 3 4

Offspring 1: 4 5 1 2 3

Offspring 2: 5 4 1 2 3

For the TSP the mutation operator must be customized, since randomly changing a bit will likely result in an invalid path. A variety of mutation operators have been employed on the chromosomes of Table 3.13, column 7:

1. Three-swap mutation: three-way swap of three randomly chosen genes in a permutation.

Chromosome 8: 3 5 2 1 4 New : 3 1 2 5 4

2. Shift mutation: a randomly selected section of a gene is moved as a block a random position in that gene.

Chromosome 10: 2 4 3 5 1 New : 2 5 1 4 3

3. Swap mutation: the values contained in two random positions are exchanged.

Chromosome 2: 5 3 2 1 4 New : 5 4 2 1 3

4. Adjacent swap mutation: it exchanges the contents of two consecutive positions within the chromosome, so it may be considered as a restricted version of the swap mutation operator, in that cut or exchange positions are always consecutive.

Chromosome 4: 2 4 3 1 5 New : 2 3 4 1 5

5. Inversion mutation: two random points within the string are selected and the segment between them is inverted.

Chromosome 6: 5 4 1 2 3 New : 5 2 1 4 3

3.3.5 Replacement

Using the elitism strategy of replacement, the best solution in the current population is added to the new population replacing the worst solution. After mutation the new population of chromosomes are evaluated using the objective function. Table 3.14 shows the elitism strategy for the GEC objective function where solution one (chromosome 1) in the new population is the worst solution, so it is replaced by the best solution in the current population.

Table 3.14: The elitism replacement strategy for the GEC objective function.

No	After mutation	$f(x) = \Gamma$	After replacement	$f(x) = \Gamma$
1	2 1 1, 2 1 1 2 1	0.2308	2 2 1, 2 2 2 2 1	0.5455
2	2 2 1, 2 2 1 2 1	0.4545	2 2 1, 2 2 1 2 1	0.4545
3	1 2 2, 1 2 2 1 1	0.2500	1 2 2, 1 2 2 1 1	0.2500
4	1 2 1, 2 1 1 2 1	0.7778	1 2 1, 2 1 1 2 1	0.7778
5	1 1 1, 1 2 2 1 1	0.3571	1 1 1, 1 2 2 1 1	0.3571
6	1 2 2, 1 2 1 2 2	0.4545	1 2 2, 1 2 1 2 2	0.4545
7	2 2 1, 2 2 2 2 1	0.5455	2 2 1, 2 2 2 2 1	0.5455
8	1 2 2, 1 1 1 1 2	0.4000	1 2 2, 1 1 1 1 2	0.4000
9	2 1 2, 2 1 2 1 2	0.4545	2 1 2, 2 1 2 1 2	0.4545
10	2 1 2, 1 2 2 2 1	0.3333	2 1 2, 1 2 2 2 1	0.3333
Standard deviation				0.1576

Table 3.15 shows the elitism strategy for the TSP objective function where solution two (chromosome 2) in the new population is the worst solution, so it is replaced by the best solution in the current population.

Table 3.15: The elitism replacement strategy for the TSP objective function.

No	After mutation	Distance function $f(x)$	$g(x) = 1/f(x)$	After replacement	Distance function $f(x)$	$g(x) = 1/f(x)$
1	3 5 2 4 1	5	0.200	3 5 2 4 1	5	0.200
2	5 4 2 1 3	11	0.090	3 5 2 1 4	4	0.250
3	5 2 1 4 3	8	0.125	5 2 1 4 3	8	0.125
4	2 3 4 1 5	9	0.111	2 3 4 1 5	9	0.111
5	4 5 1 2 3	8	0.125	4 5 1 2 3	8	0.125
6	5 2 1 4 3	8	0.125	5 2 1 4 3	8	0.125
7	5 4 1 2 3	7	0.143	5 4 1 2 3	7	0.143
8	3 1 2 5 4	6	0.167	3 1 2 5 4	6	0.167
9	4 1 2 3 5	5	0.200	4 1 2 3 5	5	0.200
10	2 5 1 4 3	8	0.125	2 5 1 4 3	8	0.125
Standard deviation						0.0366

3.3.6 Termination

The GA moves from generation to generation selecting and reproducing parents until a termination criterion is met. The most frequently used stopping criterion is a specified maximum number of generations. Another termination strategy involves population converge criteria. In general, GA will force much of the entire population to converge to a single solution. When the sum of the deviations among individuals becomes smaller than some specified threshold, the algorithm can be terminated. The algorithm can also be terminated owing to a lack of improvement in the best solution over a specified number of generations. Alternatively, a target value for the standard deviation of the evaluation measure can be established on the basis of some arbitrarily 'acceptable' threshold. Several strategies can be used in conjunction with each other. In this thesis a hybrid termination approach was adopted to check the near-optimal path. Termination parameters, which are the target standard deviation among the fitness values, and the maximum number of iterations or generations, were hybridized in order to cancel any redundant computations, hence minimize the CPU time.

The hybrid termination strategy aims to reach the target standard deviation before reaching the maximum number of generations, a function for checking the standard deviation of the current generation and comparing it with the target standard deviation (TSD) is built. If the first is less than the second, so the GA should terminate, otherwise, the system will complete GA operations until reaching the TSD or the maximum number of generations.

$$TSD = \sum_{i=1}^n \frac{(x_i - \bar{x})^2}{n - 1} \dots\dots\dots(3.10)$$

$$\eta_1 = \frac{e_d}{\sum_{r=1}^k M_r N_r} \quad \eta_2 = 1 - \left[\frac{e_o}{mn - \sum_{r=1}^k M_r N_r} \right] \dots\dots\dots(3.12)$$

where,

m: number of machines.

n: number of parts.

M_r : number of machines in the r^{th} cell.

N_r : number of parts in the r^{th} cell.

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 cell.

η : grouping efficiency.

q: weighing factor ($0 \leq q \leq 1$).

Grouping efficiency ranges from 0 to 1. Higher grouping efficiency means the more structured the solution is. In turn it means that a solution contains fewer exceptional elements. η_1 is the ratio of the number of '1s' in the diagonal blocks of the rearranged matrix to the total number of possible '1s' in all the diagonal blocks. This measure focuses on the within cell utilization or the within cell density. It is urged 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 '1s' in the off-diagonal blocks to the total number of possible '1s' in the off-diagonal blocks. This measure focuses on the intercell material handling cost. Higher value of this measure means that only a few operations are carried out in

more than one cell, therefore, maximizing this measure is similar to minimizing materials handling. Many researchers use a value of $q=0.5$, which is also used in this thesis.

Although grouping efficiency is used as a measure of the quality of solutions, it suffers some limitations. For example a very bad solution with many exceptional element often shows efficiency figures around 75% (Sarker and Mondal, 1999). Also, for large matrices, the denominator of the first term is more 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.

3.4.2 Cell index

Cell index (Mukattash, 1997) measure is considered to be new since, the efficiency of individual cells can be determined. Unlike the existing measures in the literature CI measure can distinguish between two manufacturing systems having the same sum of voids and exceptions, because cell size is taken into consideration and the matrix size is ignored. Since the CI is the sum of all individual cells, then the designer can discover the source of high and low value of goodness measure.

$$CI = \frac{1}{n} \left[\frac{k_i n_i}{k_i + v_i + e_i} \right] \dots\dots\dots(3.13)$$

Where,

n : Total number of machines in the matrix

n_i : number of machines in the i th diagonal block [ith cell]

v_i : number of voids in the i th diagonal block

e_i : number of exceptional elements in the i th off-diagonal block

k_i : number of operations in the i th diagonal block [total number of ones in the i th

cell]

p : total number of diagonal blocks [total number of cells in the matrix]

k_i+e_i : total number of operations in the matrix

Properties of the cell index measure (Mukattash, 1997):

1. Non negativity: All the elements of the cell index are positive.
2. Physical meaning of extremes. When all the ones are outside the diagonal block (condition of zero efficiency), then $CI=0$, because $k_1= k_2= k_p= 0$. On the other side, for perfect diagonal block (condition of 100% efficiency), the $CI=1$, because: $v_1= v_2 =v_p=0$, and $e_1= e_2= e_p=0$.
3. From properties 1 and 2 it is found that $0 \leq CI \leq 1$.

For the illustrative example, these grouping measures will be used to evaluate the efficiency of the block diagonal form produced in Table 3.8.

Table 3.8: Intermediate solution

	p3	p5	p2	p1	p4
m3	1	0	1	0	0
m1	1	1	1	1	0
m2	0	0	0	1	1

Grouping efficiency η :

$m=3, n=5, m_1=2, m_2=1, n_1=3, n_2=2, cd=7, eo=1, k=2$;

So,

$$\eta_1 = 7 / (2 \times 3) + (1 \times 2) = 0.875,$$

$$\eta_2 = 1 - (1 / (3 \times 5 - (2 \times 3) + (1 \times 2))) = 0.857,$$

$$\text{then } \eta = 0.5 \eta_1 + 0.5 \eta_2 = 0.86.$$

Cell index CI:

$n=3, n_1=2, n_2=1, v_1=1, v_2=0, e_1=1, e_2=0, k_1=5, k_2=2$, then

$$CI = (5 \times 2) / (5 + 1 + 1) / 3 + (2 \times 1) / (2 + 0 + 0) / 3 = 0.48 + 0.33 = 0.81.$$

The percentage of goodness for the first cell is $0.48/0.81 = 59.2\%$, and the percentage for the second cell is $0.33/0.81 = 40.7\%$. This indicates that the first cell which contains machines m_1 and m_3 is better than cell two which contains only one machine (m_2).

3.4.3 Experimental results (Graphical User Interface)

The experimental result module takes information from the GA module and displays the results on a separate module. The displayed data is:

1. grouping efficacy
2. grouping efficiency
3. TSP results
4. number of voids
5. number of exceptional elements
6. maximum number of cells
7. maximum number of generations
8. a plot of the maximum, average and standard deviation values of solutions variation with the number of generations.

3.5 Optimization of the Control Parameters for the Genetic Algorithm

Using Metalevel GA

The task of optimizing a complex system presents at least two levels of problems for the system designer. First, a class of optimizing algorithm must be chosen that is suitable for the application to the system. Second, various parameters of the optimization algorithm need to be tuned for efficiency. Even when an appropriate class of optimization

algorithm is available, there are usually various parameters that must be tuned. Often the choice of parameters can have significant impact on the effectiveness of the optimization algorithm. The problem of tuning the primary algorithm represents a secondary, or metalevel, optimization problem.

In this thesis, the genetic algorithm (GA) is used to optimize the manufacturing cell formation problem. GA is applied to the two level tasks of identifying efficient GA parameters and also to solve the manufacturing cell formation optimization problem.

That genetic algorithms can be viewed as the iterated probabilistic application of a set of operators on a population, and that in most cases these operators are static that is to say that their forms and parameters are fixed and apply uniformly to the whole population i.e. they are global. Population-level adaptation algorithms can be typified as using a fixed set of global operators, but allowing their parameters to vary over time. The most important parameter is of course the probability of application.

We now describe how to optimize the performance of GA whose search space is defined by four control parameters (population size, target standard deviation, crossover probability, and mutation probability), and to identify the optimal parameter settings. GA performed the searches for the optimal GA parameters, which demonstrates the efficiency and power of GA as metalevel optimization techniques. A metalevel GA could similarly search any other space of parameterized optimization procedures.

3.5.1 The space of the genetic algorithms

This study is limited to a particular subclass of GA's characterized by the following four parameters:

1. Population size (n): the population size affects both the ultimate performance and the efficiency of GA. GA generally does poorly with very small populations, because the population provides an insufficient sample size for most hyperplanes. A large population is more likely to contain representatives from a large number of hyperplanes. Hence, the GA can perform a more informed search. As a result, a large population discourages premature convergence to suboptimal solutions. On the other hand, a large population requires more evaluations per generation, possibly resulting in an unacceptably slow rate of convergence. In the current experiments, the population size ranged from 10 to 160 in increments of 10.
2. Crossover Rate (P_c): the crossover rate controls the frequency with which the crossover operator is applied. In each new population, $P_c \times n$ structures undergo crossover. The higher the crossover rate, the more quickly new structures are introduced into the population. If the crossover rate is too high, high-performance structures are discarded faster than selection can produce improvements. If the crossover rate is too low, the search may stagnate (find local optima) due to the lower exploration rate. The current experiments allowed 16 different crossover rates, varying from 0.25 to 1.00 in increments of 0.05.
3. Mutation Rate (P_m): mutation is a secondary search operator which increases the variability of the population. After selection, each bit position of each structure in the new population undergoes a random change with a probability equal to the mutation rate P_m . Consequently, approximately $P_m \times n \times L$ mutations occur per generation where L is the chromosome length. A low level of mutation serves to prevent any given bit position from remaining forever converged to a single value in the entire population. A high level of mutation yields an essentially random search. The current experiments allowed mutation rate to change from 0.0 to 1.0 exponentially.

4. Target standard deviation (TSD): the most frequently used stopping criterion is a specified maximum number of generations. The algorithm can also be terminated owing to a lack of improvement in the best solution over a specified number of generations. Alternatively, a target value for the evaluation measure can be established on the basis of some arbitrarily 'acceptable' threshold. In this thesis, termination parameters, which are the target standard deviation (TSD), and the maximum number of iterations or generations, were hybridized to check the near-optimal path. The current experiment allowed 10 different values varying from 0.005 to 0.05 in increments of 0.005.

We denote a particular GA by indicating its respective values for the parameters GA (n , TSD, P_c , P_m). Early work by De Jong (Grefenstette, 1986) suggests parameter settings which have been used in a number of implementations of genetic algorithms. He define the standard GA as $GAs = GA(50, 0.6, 0.001)$, so this was the natural choice for the metalevel GA.

In some cases, it is possible to predict how variations of a single parameter will affect the performance of the GA's assuming that all other parameters are kept fixed. However, it is difficult to predict how the various parameters interact. For example, what is the effect of increasing the population size, while lowering the crossover rate?

A common approach of metalevel GA is to adjust one or more parameters dynamically in accordance with the performance of the algorithm or some measured quantity of the population. A well-known and popular approach is to keep statistics on the performance of offsprings generated by various reproductive operators relative to their parents (Smith and Fogarty, 1997). Periodically, *successful* operators, which produced the best solution of

the GA, are rewarded by increasing their probability of application relative to less successful operators. This approach does however require extra memory, since it is usually found necessary to maintain family trees of the operators which led to a given individual, in order to escape from local optima.

3.6 Summary

In this chapter a complete discussion of the manufacturing cell formation problem is presented. The genetic algorithm methodology which uses roulette wheel selection, two objective functions, a combination of crossover and mutation techniques, elitism replacement and hybrid termination criteria is described in details. This chapter also presents three methods to evaluate the efficiency of the block diagonal form. In addition, a new way of optimizing the GA parameters is described where metalevel GA is used to solve the two levels of the manufacturing cell formation problem.

4- EXPERIMENTAL RESULTS

4.1 The Programming Language: MATLAB

MATLAB is becoming increasingly popular among students, researchers, technicians and engineers because of MATLAB features such as immediate graphing facilities, built-in functions, the possibility of adding user-written functions and simple programming. The package provides useful tools for interfacing with external programs and data sets, as well as options for keeping records of calculations which can be later transformed into technical reports. The versatility of the basic MATLAB package can be enhanced by separately-available softwarees designed for specialized, and advanced fields of application (Biran and Breiner, 1999).

MATLAB is a technical computing environment for high-performance numeric computation and visualization. MATLAB integrates numerical analysis, matrix computation, signal processing, and graphics in an easy-to-use environment where problems and solutions are expressed just as they are written mathematically- without traditional programming.

The name MATLAB stands for MATrix LABoratory. It was originally written to provide easy access to matrix software. MATLAB is an interactive system whose basic data element is a matrix that does not require dimensioning. This allows you to solve many numerical problems in a fraction of the time it would take to write a program in a language such as Fortran, Basic, or C.

MATLAB has evolved over a period of years with input from many users. Typical uses included general purpose numeric computation, algorithm prototyping, and special purpose problem solving with matrix formulations.

MATLAB also features a family of application-specific solutions that we call softwarees. Very important to most users of it, softwarees are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment in order to solve particular classes of problems. May be the most important feature of MATLAB, and one that we took care to perfect, is its easy extensibility. This allows you to become a contributing author too, creating your own applications.

In this thesis MATLAB is used to construct a comprehensive software which solves the manufacturing cell formation problem using the genetic algorithm.

The main characteristics of this software are the following:

- Hierarchical structure
- The flexibility to choose between two solving methodologies:
 1. Using the simple GA to solve the MCFP after specifying these parameters: population size, target standard deviation and mutation and crossover rates.
 2. Using metalevel GA where the control parameters (population size, target standard deviation and mutation and crossover rates) are determined by the first level GA and used to solve the second level GA, which gives the cell formation problem solutions.
- Ability to switch between two objective functions (grouping efficacy (GEC) ,and travelling salesman problem formulation (TSP)).
- Flexibility to specify the maximum number of cells (k_{max}) so that, the solved matrix should be partitioned into a number of cells less or equal to this number.
- Hybrid termination strategy (target standard deviation, and maximum number of generations).

The reader can find the computer implementation of the manufacturing cell formation problem in Appendix A. It describes the MATLAB software and its flexible and efficient characteristics and gives the details about the most important internal functions and graphical user interfaces. In addition, Appendix B presents the different screens of the MATLAB software.

4.2 General Observations of the Metalevel Genetic Algorithm

This experiment identified $GA_{best} = GA(30, 0.005, 0.95, 0.01)$. The performance improvement between standard GAs (50, 0.6, 0.001) and GA_{best} can be attributed to an interaction among a number of factors. For example, GA_{best} uses a smaller population, which allows many more generations within a given number of trials where, GA_{best} iterated through an average of twice as many generations as GAs. The smaller population size is apparently balanced by the significantly increased mutation rate and crossover rate in GA_{best} . A higher crossover rate tends to disrupt the structures selected for reproduction at a high rate, which is important in a small population, since high performance individuals are more likely to quickly dominate the population. The higher mutation rate also helps prevent premature convergence to local optima.

On the other hand, if a very small population size is used (i.e., $n = 10$), the number of representatives from any given hyperplane is so small that the selection procedure has insufficient information to properly apportion credit to the hyperplanes represented in the population. As a result, a relatively good structure may overrun the entire population in a few generations. Unless the mutation rate is high, the GA will quickly converge to a suboptimal solution. In contrast, random search will usually locate at least one high performance point within the first thousand trials, leading to performance. That is,

random search is fairly though competitor for search strategies when the population size is small.

With a larger population and higher mutation rate, the population will tend to contain more variety, thus increasing the random aspects of the GA i.e. tends to reduce the effects of selection, resulting in a less focused search. The lower crossover balances these aspects, which tend to enhance the selective pressure.

Mutation rates above 0.05 are generally harmful, with performance approaching that of random search, with rates above 0.1 regardless of the other parameter settings. The absence of mutation is also associated with poorer performance, which suggests that mutation performs an important service in refreshing lost values. Best performance can be obtained with a population size in the range of 30-100 structures.

The performance data also suggests other regularities, for example, in small populations (20 to 40 structures), good performance is associated with either a high crossover rate combined with a low mutation rate or a low crossover rate combined with a high mutation rate. For mid-sized populations (30 to 90 structures), the optimal crossover rate appears to decrease as the population size increases. This is reasonable since, in smaller populations, crossover plays an important role in preventing premature convergence. In summary, the performance of GA's appears to be a nonlinear function of the control parameters. However, the available data is too limited to confirm or disconfirm the existence of discontinuities or multiple local optima in the performance space of GA.

4.3 Experimental Results

The MATLAB software is tested on ten data sets (see Table 4.1) from the literature to study its effectiveness as a clustering tool. To maximize cell design flexibility, an implementation was adopted that allows the user to employ various representation schemes, sets of operators, and evaluation measures. This flexibility is gained through the MATLAB software.

Table 4.1: Experiments Data Sets (Test Problems)

No	Incidence matrix	Reference	Size	e
1	King (1980a)	Mukattash (1997)	6x10	25
2	Chen and Guerrero (1994)	Mukattash (1997)	6x15	40
3	Chu and Hayya (1991)	Zhao et al. (1995)	9x9	32
4	King (1980b)	Mukhopadhyay et al. (2000)	10x8	25
5	Viswanathan (1996)	Mukattash (1997)	10x12	41
6	Simple Chan and Milner (1982)	Malave and Ramachandran, (1994)	15x10	46
7	Complex Chan and Milner (1982)	Malave and Ramachandran, (1994)	15x10	49
8	Chandrasekharan and Rajagopalan (1986)	Cheng et al. (1998)	8x20	61
9	King (1980c)	Malave and Ramachandran, (1994)	14x24	61
10	Burbidge (1960)	Jeffrey et al. (1996)	20x35	136

e: total number of ones (operations) in the incidence matrix.

For data sets taken from the literature, the maximum number of permissible cells, k_{max} , was initially set equal to the best known number of cell, k^l , as determined by other cell formation algorithms. The maximum number of generations needed is dependent on the size of the problem. Because GA is stochastic search algorithm, the number of generations needed to solve a particular problem also depends on the composition of the initial population. Figure 4.1 shows the great improvement on the solution of the GA, from the first generation till the generation, which gives the best solution. Note the great improvement in the best solution which changed from 0.38 to 0.86 in 34 generations i.e., 126.3 % improvement.

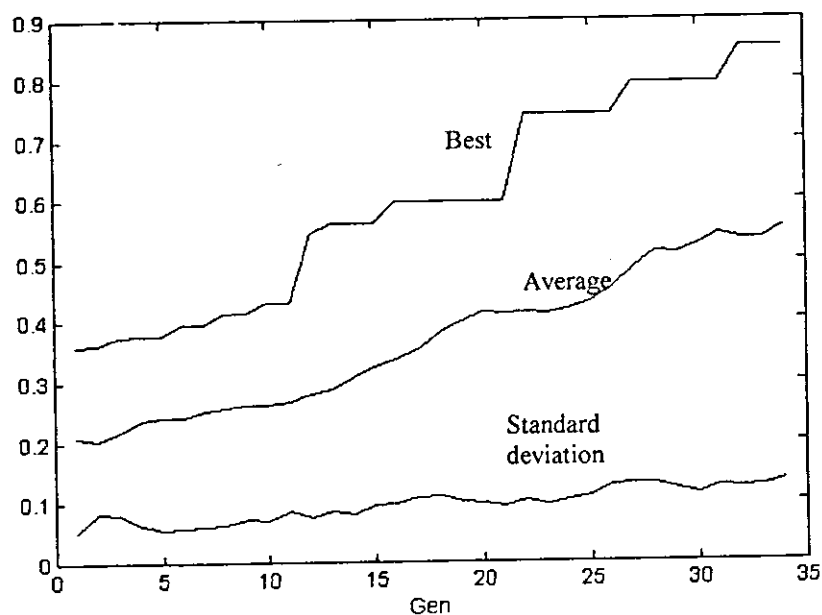


Figure 4.1 The best, average and standard deviation variation with the number of generations.

The data sets used to evaluate this clustering technique are of varying size and complexity. In each case, correct grouping is defined as being equivalent to the best known configuration provided in the literature, on the basis of grouping efficacy and grouping efficiency. Table 4.2a summarizes the experimentation results for the ten sample data sets. In all cases, the simple GA was able to determine a configuration with grouping efficacy measure equal to any previously reported results (except for problem 2 and 9). On the other hand, Table 4.2b summarizes the results of using the metalevel GA to solve the data sets. It is clear that these results are much better than those of the simple GA, where data sets numbered one, three, five and nine results outperforms those of the simple GA and even the best values found in the literature. This right except for data set two, this is because it could happen that GA could theoretically solve a certain problem, but it will not return the correct answer. This is because of the random nature of the algorithm and its reliance on natural selection.

Table 4.2a: The experimentation results for the 10 sample data sets using simple GA.
 k^L : optimal no. of cells presented in the literature, k_{max} : optimal no. of cells achieved by simple GA.

Data set	Benchmark				Simple GA results							
	No.	k^L	Γ	η	CI	Γ	η	CI	TSP Parts	TSP machines	e_o	e_v
1	2	71.1	85.0	70.3	71.9	85.0	70.3	16	18	2	7	2
2	2	64.1	NA	63.7	60.4	76.7	58.6	38	38	8	13	3
3	3	73.5	89.8	72.4	73.5	89.8	72.4	28	27	7	2	3
4	3	82.2	92.5	81.9	82.2	92.5	81.9	20	18	2	3	3
5	3	58.9	78.9	58.0	58.9	78.9	58.0	38	45	8	15	3
6	3	92.0	96.0	92.8	92.0	96.0	92.8	22	16	0	4	3
7	3	85.4	94.5	86.1	85.4	94.5	86.1	25	26	5	3	3
8	3	85.2	95.8	84.4	85.2	95.8	84.4	38	30	9	0	3
9	4	51.7	77.6	59.5	50.0	74.3	52.4	58	58	1	59	3
10	4	75.7	88.1	75.5	75.7	88.1	75.5	67	90	2	41	4

Table 4.2b: The experimentation results for the 10 sample data sets using metalevel GA.
 k^L : optimal no. of cells presented in the literature, k_{max} : optimal no. of cells achieved by metalevel GA.

Data set	Benchmark				metalevel GA results							
	No.	k^L	Γ	η	CI	Γ	η	CI	TSP Parts	TSP machines	e_o	e_v
1	2	71.1	85.0	70.3	72.4	86.3	68.6	13	17	4	4	3
2	2	64.1	NA	63.7	61.7	80.1	61.8	30	34	11	7	3
3	3	73.5	89.8	72.4	74.3	89.1	70.4	21	22	6	3	3
4	3	82.2	92.5	81.9	82.2	92.5	81.9	20	18	2	3	3
5	3	58.9	78.9	58.0	59.6	80.5	58.3	33	40	10	11	3
6	3	92.0	96.0	92.8	92.0	96.0	92.8	22	16	0	4	3
7	3	85.4	94.5	86.1	85.4	94.5	86.1	25	26	5	3	3
8	3	85.2	95.8	84.4	85.2	95.8	84.4	38	30	9	0	3
9	4	51.7	77.6	59.5	61.8	81.4	62.4	50	56	3	31	4
10	4	75.7	88.1	75.5	75.7	88.1	75.5	67	90	2	41	4

The computational times depends on many factors such as the number of generations the permissible number of cells and of course, the size and complexity of the initial matrix. They ranged from two to forty minutes. The problem complexity may be affected by the number of exceptional elements in the machine-part incidence matrix. For example, problems number 6 and 7 have the same number of parts and machines. Problem 6 has no exceptional elements while problem 7 has 5 exceptional elements after grouping. The computational times for the two problems (3.1 minutes for problem 6 and 3.88 minutes for problem 7) have, to a certain extent reflected the complexity of the

problems. The cell formations for selected data sets are shown in Tables 4.3 to 4.6. The simple Chan and Milner (1982) data set number 6, taken from Malave and Ramachandran, (1994), is a 15 x 10 problem containing four voids and no exceptional elements (no bottleneck machines) and is shown with its solution in Tables 4.3 and 4.4. The next data set which is Burbidge, (1960) 20 x 35 incidence matrix taken from (Jeffrey *et al*, 1996) is shown in Table 4.5. The solution for this data set which contains two exceptional elements i.e. two bottleneck machines and 41 voids, is displayed in Table 4.6.

Table 4.3: The original incidence matrix for 15 x 10 Simple Chan and Milner problem (1982), data set number 6.

		Parts									
		1	2	3	4	5	6	7	8	9	10
Machines	1	0	0	1	1	0	1	0	0	0	0
	2	1	0	0	0	0	0	1	0	0	1
	3	0	1	0	0	1	0	0	1	0	0
	4	0	0	0	1	0	1	0	0	1	0
	5	0	1	0	0	1	0	0	1	0	0
	6	0	0	1	0	0	1	0	0	1	0
	7	0	0	0	0	0	0	1	0	0	1
	8	0	1	0	0	1	0	0	1	0	0
	9	0	0	1	1	0	1	0	0	1	0
	10	1	0	0	0	0	0	1	0	0	1
	11	1	0	0	0	0	0	1	0	0	1
	12	1	0	0	0	0	0	1	0	0	1
	13	0	1	0	0	1	0	0	1	0	0
	14	0	0	1	1	0	1	0	0	1	0
	15	1	0	0	0	0	0	1	0	0	1

Table 4.4: Metalevel GA solution for the 15 x10 Simple Chan and Milner problem (1982), data set number 6. $\Gamma=92.0$ $\eta=96.0$ $CI=92.8$

		Parts									
		5	2	8	1	7	10	3	9	6	4
Machines	13	1	1	1	0	0	0	0	0	0	0
	3	1	1	1	0	0	0	0	0	0	0
	8	1	1	1	0	0	0	0	0	0	0
	5	1	1	1	0	0	0	0	0	0	0
	7	0	0	0	1	1	1	0	0	0	0
	2	0	0	0	0	1	1	0	0	0	0
	10	0	0	0	1	1	1	0	0	0	0
	12	0	0	0	1	1	1	0	0	0	0
	15	0	0	0	1	1	1	0	0	0	0
	11	0	0	0	1	1	1	0	0	0	0
	1	0	0	0	0	0	0	1	1	1	0
	4	0	0	0	0	0	0	0	1	1	1
	14	0	0	0	0	0	0	1	1	1	1
	9	0	0	0	0	0	0	1	1	1	1
	6	0	0	0	0	0	0	1	0	1	1

Table 4.6: Metalevel GA solution for the 20 x 35 Burbidge problem (1960), data set number 10. $\Gamma=75.8$ $\eta=88.1$ $CI=75.5$

		Parts																																						
		1	3	5	15	17	20	23	25	29	4	6	9	11	21	28	30	32	33	35	2	7	10	12	13	18	24	27	31	8	14	16	19	22	26	34				
Machines	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	3	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0		
	7	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	8	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	17	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	11	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	12	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	19	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4.3.1 The importance of the cell index as a measure of the goodness of grouping solutions

As mentioned in section 3.4.2, the cell index can distinguish between two manufacturing systems having the same sum of voids and exceptions, because cell size is taken into consideration. Metalevel GA is used to solve King 14 x 24 (1980c) problem which is taken from (Malave and Ramachandran, 1994). Two alternative solutions are produced with the same sum of voids and exceptions, their results are given in Table 4.7. The original matrix and its two alternative solution matrices are given in Appendix C. The results show that the two solutions have the same number of cells, which is four, the same part families and the same sum of voids and exceptions. The only difference is the machines contained in machine cells two and three, where machine number seven is taken out of cell two in solution one and placed in cell three in solution two. As noted grouping efficacy and grouping efficiency are unable to detect this difference significantly, whereas cell index measure indicates that solution two is more efficient. This also, proves the ability of the metalevel GA to give alternative solutions to the cell designer, which increases its flexibility as a cell formation technique.

Table 4.7: Alternative solutions with the same sum of voids and exceptions for the 14 x 24 King (1980c) problem.

Solution 1	eo=4, ev=30, eo+ev=34, $\Gamma=61.36$, $\eta=81.35$, CI=61.20	
	Machines	Parts
Cell 1	2, 3, 10, 11	3, 4, 21, 24
Cell 2	1, 7, 12, 13	6, 7, 8, 18
Cell 3	4, 5	1, 2, 17, 19, 20, 23
Cell 4	6, 8, 9, 14	5, 9, 10, 11, 12, 13, 14, 15, 16, 22.
Solution 2	eo=3, ev=31, eo+ev=34, $\Gamma=61.79$, $\eta=81.38$, CI=62.43	
	Machines	Parts
Cell 1	2, 3, 10, 11	3, 4, 21, 24
Cell 2	1, 12, 13	6, 7, 8, 18
Cell 3	4, 5, 7	1, 2, 17, 19, 20, 23
Cell 4	6, 8, 9, 14	5, 9, 10, 11, 12, 13, 14, 15, 16, 22.

4.3.2 Effects of constraining the number of permissible cells

The metalevel GA using the TSP objective function can not predict the effect of varying the maximum number of cells on the formed machine cells and part families. This effect is obviously included in the GEC objective function because of its special type of chromosome representation. The solutions generated by GEC objective function are extremely dependent on the priori specification of the desired number of cells.

Specifying a maximum number of cells for the GA simply places an upper bound on the number of cells. Recalling that the GA is an optimization tool, it searches for the maximum grouping efficacy value under this constraint. Let k_{max} be the maximum number of cells permitted and k^* be the optimal number of cells. If $k_{max} \geq k^*$, then in the limit and in practice, the GA will find k^* cells. In effect, the upper bound constrain k_{max} is not binding and has no effect on the number of cells formed or the grouping efficacy value. However, if $k_{max} < k^*$, the constraint is binding, and the genetic algorithm will produce a solution with, at most, k_{max} cells at a reduced grouping efficacy value.

Because the optimal number of cells k^* is generally unknown, experimentation focused on comparing the GA solution to the best known number of the cells from the literature, k^l , as measured by grouping efficacy. Several experiments using the Chan and Milner problems were conducted to demonstrate the effects of k_{max} as a constraint. Tables 4.8 and 4.9 show the results of experiments with the metalevel GA when $k_{max} > k^l$, i.e., the maximum permissible number of cells exceeds the number of cells in the best known solution from the literature. For the two Chan and Milner problems (data sets numbers 6 and 7), the maximum number of cells specified was $k_{max}=4$ and $k_{max}=5$, ($k^l=3$). The results indicate that the GA solution of three cells is the appropriate number of cells. As expected, no members are assigned to the additional $k_{max}-k^l$ cell(s).

Table 4.8: Solutions to the Simple Chan and Milner (1982),
 problem number 6 ($k_{max} > k^*$ ($\Gamma=92.0$ $\eta=96.0$)).

Cell/Family	Machines	Parts
$K_{max} = 4$ and 330 generations		
Cell/Family 1	13,3,8,5	5,2,8
Cell/Family 2	7,2,10,12,15,11	1,7,10
Cell/Family 3	1,4,14,9,6	3,9,6,4
Cell/Family 4	none	None
$K_{max} = 5$ and 1000 generations		
Cell/Family 1	13,3,8,5	5,2,8
Cell/Family 2	7,2,10,12,15,11	1,7,10
Cell/Family 3	1,4,14,9,6	3,9,6,4
Cell/Family 4	none	None
Cell/Family 5	none	None

Table 4.9: Solutions to Complex Chan and Milner (1982),
 problem number 7 ($k_{max} > k^*$ ($\Gamma=85.4$ $\eta=94.5$)).

Cell/Family	Machines	Parts
$K_{max} = 4$ and 400 generations		
Cell/Family 1	14, 3, 4, 1, 6	3, 4, 8, 9
Cell/Family 2	15, 13, 10, 9, 8, 5	2, 5, 10
Cell/Family 3	12, 11, 2, 7	1, 6, 7
Cell/Family 4	none	none
$K_{max} = 5$ and 1200 generations		
Cell/Family 1	14, 3, 4, 1, 6	3, 4, 8, 9
Cell/Family 2	15, 13, 10, 9, 8, 5	2, 5, 10
Cell/Family 3	12, 11, 2, 7	1, 6, 7
Cell/Family 4	none	none
Cell/Family 5	none	none

As the value of k_{max} increases, the state space the GA must explore also grows, thus increasing, on average, the number of generations required. In practice, the number of generations required increases at a much slower rate than the growth of the space. Figure 4.2 shows this phenomena for the Simple Chan and Milner problem over a range of k_{max} from 3 to 5. Values in the graph represent the number of unique solutions in the state space for each value of k_{max} .

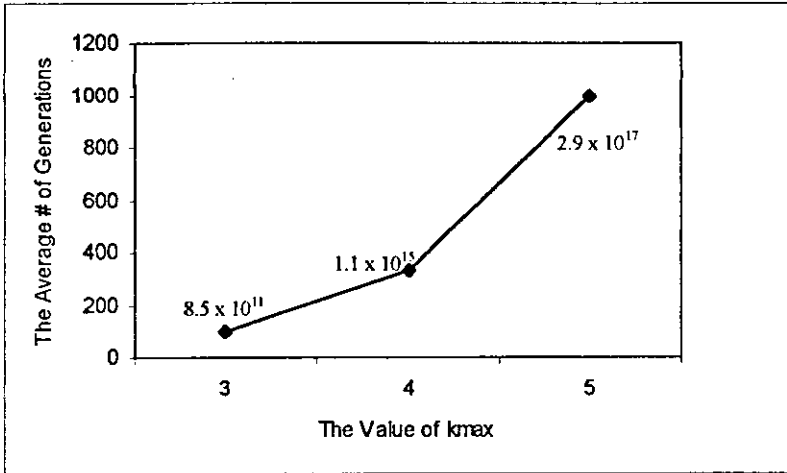


Figure 4.2: Average number of generations versus selected values of k_{max} .

A cell designer might wish to begin with k_{max} set to the maximum number of cells believed practicable for the facility. In subsequent analysis, the value of k_{max} could be progressively restricted to reflect limits arising from material handling constraints or efforts to achieve effective team work.

To further analyze constrained aggregation, the Simple Chan and Milner problem, for which $k^I=3$, is examined. Given $k_{max} = 2$, the GA could produce four possible configurations; any one of three different two-cell configurations or a single cell. Table 4.10 displays the grouping efficacy for each of the four possibilities for the Simple Chan and Milner problem. The two cells that produce the minimum number of voids when combined achieve the highest grouping efficacy. In this way, the GA determined the optimal configuration under the binding constraint $k_{max}=2$.

Table 4.10: Comparison of the alternative configurations for Simple Chan and Milner problem ($k_{max}=2$.)

Combined	e_o	e_v	Γ
1 and 2	0	34	57.50
1 and 3	0	35	56.79
2 and 3	0	43	51.69
All 3	0	86	30.67

4.3.3 Comparing the GEC and the TSP results

The GEC and the TSP solutions to seven of the ten data sets are the same. But this result can not be generalized. For example, this result is not applicable for the data set shown in Table 4.11, where the GEC and the TSP solutions are not the same. See Table 4.12 and 4.13.

Table 4.11: Sample data set

		Parts									
		1	2	3	4	5	6	7	8	9	10
Machines	1	1	1	1	0	0	0	0	0	0	0
	2	1	1	1	0	0	0	1	0	0	0
	3	0	0	0	1	0	0	0	0	0	1
	4	0	0	0	1	0	0	0	0	0	1
	5	0	0	0	1	1	1	0	1	1	1

Table 4.12: The solution matrix for the sample data set using GEC objective function.
 $eo=2, ev=1, \Gamma=83.3, \eta=93.9, CI=85.0$

		Parts									
		1	2	3	7	4	10	5	6	8	9
Machines	1	1	1	1	0	0	0	0	0	0	0
	2	1	1	1	1	0	0	0	0	0	0
	3	0	0	0	0	1	1	0	0	0	0
	4	0	0	0	0	1	1	0	0	0	0
	5	0	0	0	0	1	1	1	1	1	1

Table 4.13: The solution matrix for the sample data set using TSP objective function.
 $eo=0, ev=9, \Gamma=65.4, \eta=82.7, CI=68.3$

		Parts									
		4	10	5	6	8	9	7	1	2	3
Machines	5	1	1	1	1	1	1	0	0	0	0
	3	1	1	0	0	0	0	0	0	0	0
	4	1	1	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	1	1	1
	2	0	0	0	0	0	0	1	1	1	1

The two fitness functions give different configurations because each one has its own objective. For the GEC the objective is to minimize the sum of voids and exceptions ($eo+ev = 3$), while the TSP objective is to minimize the number of exceptions ($eo= 0$).

4.4 Comparison with the Literature

The cell formation methodology used in this thesis offers several advantages over existing methods. It simultaneously groups parts and machines into families and cells, and eliminates the need for visual analysis of the solution. It also offers improved flexibility by allowing the cell designer to use various objective functions and incorporate design constraints during cell formation. These capabilities allow alternative cell configuration to be generated and reviewed easily. These advantages in addition to the GA control parameters optimization using the metalevel GA, are some points that can be utilized to compare the work of this thesis with the previous work.

Mukhopadhyay et al. (2000), used a graph theoretic approach to group machines into cells and parts into families based on the Modified Hamiltonian Chain. His approach gives the machine sequence and the part sequence directly, so the incorporation of constraints on the number of cells is not allowed.

Mukattash (1997) derived a modified approach for solving the p-median problem in Group Technology by specifying the number of cells in advance and constraining the cell size. The output of this approach are cells of machines, while parts are then assigned to these cells so as to minimize the number of voids and or the number of exceptions. In our algorithm using GEC objective function, the machine cells and part families are grouped concurrently.

In this thesis, part family formation problem is solved under the assumption that, the machine or operation requirements are the part family differentiating attributes. Lee-Post (2000), used a simple genetic algorithm to form part families based on a classification and coding system. She used a coding system, which uses five attributes to describe a part:

shape, features, size, precision and material. So, part families are formed based on the similarities in these attributes. It is evident that a different technique is required to form machine cells based on the grouped part families, which increases the problem complexity.

An alternative approach to the optimization of the GA control parameters is to allow the GA to modify its own parameters dynamically during the search. Dimopoulos and Mort (2001) used the genetic programming (GP) algorithm to achieve optimized control parameters to solve the cell formation problem. They proposed that the operation of GP is similar to any other evolutionary computation method in the sense that the Darwinian principle of strife for survival is employed during the search for an optimal solution. However, while most evolutionary algorithms evolve fixed-length strings of binary, integer or real numbers, genetic programming evolves solutions in the form of computer programs of variable length. However, for many optimization problems the number of evaluations, which can be performed in a reasonable amount of time and the problem complexity would not allow the GP enough evaluations to modify its search techniques to any significant degree. Therefore the metalevel are important in that it identifies approximately optimal parameter settings for the performance measure considered.

4.5 Summary

This chapter gives a comparison between the results produced by simple GA and the metalevel GA, where the results proved that the metalevel GA almost always get better solutions to the manufacturing cell formation problem. Also, the importance of the cell index as a measure of the goodness of grouping solution is emphasized. In addition, the effect of constraining the number of permissible cells is examined in details. Finally, a comparison with the literature is presented.

5- SUMMARY, CONCLUSIONS
AND
RECOMMENDATIONS

5.1 Summary

The manufacturing cell formation problem (MCFP) has been studied in this thesis. A metalevel genetic algorithm approach has been employed to solve the two levels of the MCFP, where the first level optimizes the GA control parameters and the second level uses these parameters to solve the MCFP.

The metalevel utilizes two objective functions each one is associated with its own representation, and crossover and mutation techniques. The two objective functions used are the grouping efficacy (GEC) and the Travelling Salesman Problem (TSP) formulation. The grouping efficacy uses one crossover technique, which is the simple crossover and two mutation techniques, which are uniform and boundary mutation. On the other hand, the TSP uses three crossover techniques: single point, partially mapped, and order based crossover and five mutation techniques: adjacent swap, three-swap, shift, inversion and insertion mutation.

Ten data sets are adopted from literature to test the metalevel GA. Using the grouping efficacy alternative solutions can be achieved through the specification of the desired number of cells. These solutions and the solutions of the TSP are evaluated using three grouping measures: grouping efficiency, cell index in addition to the grouping efficacy.

In this research, a comprehensive MATLAB software is constructed to solve the MCFP. The hierarchical structure is the most important characteristic of this software, which consists of functions and graphical use interfaces.

The genetic algorithm technique uses roulette wheel selection, a combination of crossover operators, a combination of mutation operators, elitism replacement strategy and hybrid termination criteria.

The solutions for the ten data sets, which are obtained from the MATLAB software, are analyzed in details and compared with those found in the literature. These solutions are at least equal to any previously reported results. In addition the metalevel GA used in this thesis to solve the MCFP is compared with the algorithms and methodologies found in the literature.

Finally, conclusions and recommendations are discussed in details

5.2 Conclusions

A MATLAB software has been developed that uses metalevel genetic algorithm to solve the cell formation problem. The method generates quality solutions and provided improved flexibility for analyzing the cell formation problem.

The objective was to minimize the intercellular movements, since this will lead to reduce material handling cost, simplify planning and scheduling, reduce work-in-process buffer space and decrease throughput time.

The cell formation methodology used in this thesis offers several advantages over existing methods. It simultaneously groups parts and machines into families and cells, and offers improved flexibility by allowing the cell designer to use various objective functions and incorporate design constraints during cell formation. These capabilities allow alternative cell configuration to be generated and reviewed easily.

Visual identification of machine cells and part families for large matrices with a reasonable number of bottleneck machines would be very difficult using the final matrix. This is particularly so when a strictly block diagonal form is not achieved. This is not the case in the two objective functions used here, where the GA produces the cells and families concurrently.

The unique representation used in the grouping efficacy objective function effectively reduces the state space by incorporating the constraints into the variable formulation. A standard integer programming solution technique cannot be employed without significant modification because of the objective function's inability to decode this variable representation. However, for the genetic algorithm, the objective function is a computer procedure that can easily decode and evaluate a solution.

Then, the GA approach allows the designer to incorporate or selectively remove constraints on the number of permissible cells. Unconstrained solutions containing the naturally occurring clusters can be generated as well as constrained solutions. By allowing alternative cell configuration evaluation, the power of the technique as a tool of analysis is extended.

The performance of the GA methodology was compared with several clustering techniques found in the literature, where the goodness of grouping of the obtained solutions is evaluated using three measures: grouping efficiency, cell index and grouping efficacy. The cell index improved its ability to significantly distinguish between alternative solutions with the same sum of voids and exceptions.

The metalevel GA model found the 'best known' or better configurations in all the ten data sets used during experimentation, proving it to be an effective, as well as flexible, clustering technique. The best solution found in the literature for data set number two was not achieved using this algorithm, this is because it could happen that GA could theoretically solve a certain problem, but it will not return the optimal answer. This is because of the random nature of the algorithm and its reliance on natural selection, mutation and crossover. Naturally, it could happen that a certain flow of events that would lead to a correct solution will not occur and thus a solution will not be found. However, by using several unrelated populations we have decreased the probability of this happening, since if some population has poor individuals the solution could still be found at another.

The two objective functions used in this thesis are: grouping efficacy (GEC) and Travelling salesman problem formulation (TSP) are compared as follows:

Similarities between TSP and GEC:

1. Because they explicitly assigns machines and parts to cells, no ambiguity exists in determining the cell to which bottleneck machines are assigned.
2. The solutions forced the clusters to form along the diagonal block, simultaneously grouping parts and machines.

Differences between TSP and GEC:

1. Each objective function has its own unique representation. The GEC uses float representation, which allows the user to incorporate or selectively remove constraints on the number of cell, but this representation makes the GA more susceptible to local minima. While TSP uses permutation representation, which enables the GA to escape from local minima, but does not incorporate constraints on the number of cells. In addition each has its objective and gives different configuration.

2. Many genetic operators were developed to improve the performance of the GA, that take advantage of the special structure of the cellular manufacturing formation problem. Each objective function has its own crossover and mutation techniques: the grouping efficacy uses one crossover technique, which is the simple crossover and two mutation techniques, which are uniform and boundary mutation. On the other hand, the TSP uses three crossover techniques: single point, partially mapped, and order based crossover and five mutation techniques: swap, three-swap, shift, inversion and insertion mutation.

5.3 Recommendations for Further Extensions

The manufacturing cell formation problem is studied in this thesis. Research in this area is still developing and a number of potentially valuable enhancements to this work should include:

1. The application of multi-objectives evaluation function by considering the MCFP as multi-criteria decision-making problem. Objectives such as the minimization of costs of space usage, part subcontracting, machine loading and operation costs, or the minimization of the amount of part dissimilarities and machine duplications can be formulated to solve the MCFP. Most researchers when dealing with cell formation they assume an infinite loading capacity, but from practical point of view it is important to incorporate machine loading into the model. Also, to achieve higher level of production flexibility, it becomes necessary to consider alternative process plans for each part.
2. The addition of constraints that place upper bounds on cell size i.e. , the number of machines in each cell needs to be limited. Thus, a minimum and a maximum number of machines that allowed in each cell is pre-specified. This aims to simplify management control and material coordination.

3. Since GA is a randomized algorithm, the performance of GA during a single trial in the metalevel experiments represents a sample from a distribution of performances. Therefore, it is recommended that the GA showing the best solution should be subjected to more extensive testing.
4. In this thesis the number of chromosomes which are going to be crossed over or mutated are partitioned equally between crossover and mutation combined techniques. But more better solutions for the MCFP may be obtained by optimizing the ratios of these chromosomes by the metalevel GA. Also, the development of improve genetic algorithms techniques (operators) is recommended.

6 References:

Biran, A. and Breiner, M. 1999. *Matlab 5 for engineers*, 1st Edition. Addison-Wesley Longman Limited.

Cheng, C. Gupta, Y. Lee, W. and Wong, K. 1998. A TSP-Based Heuristic for Forming Machine Groups and Part Families. *International Journal of Production Research*, 36 (5): 1325-1337.

Chunwei, Z. and Zhiming, W. 2000. A Genetic Algorithm For Manufacturing Cell Formation With Multiple Routes And Multiple Objectives. *International Journal of Production Research*, 38 (2): 385-395.

Dimopoulos, C. and Mort, N., 2001. A Hierarchical Clustering Methodology Based on Genetic Programming for the Solution of Simple Cell-Formation Problems. *International Journal of Production Research*, 39 (1): 1-19.

Grefenstette, J. 1986. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16 (1): 122-128.

Grefenstette, J. Gopal, R. Rosmaita, B. and Gucht, V. 1985. Genetic Algorithms for the Travelling Salesman Problem. *International Conference on Genetic Algorithms and Their Applications*, pp. 160-168, by Lawrence Erlbaum Associates, Inc.

Goldberg, D. 1989. *Genetic Algorithms In Search, Optimization And Machine Learning, 1st Edition*, Addison-Wesley, Reading, MA, USA.

Goldberg, D. and Lingle, R. 1985. Alleles, Loci and the Travelling Salesman Problem. *International Conference on Genetic Algorithms and Their Applications*, pp. 154-159, by Lawrence Erlbaum Associates, Inc.

Jeffrey, A. Joines, C. Thomas, G. and Russell, E. 1996. Manufacturing Cell Design: an Integer Programming Model Employing Genetic Algorithms. *IIE Transactions*, 28 (5): 69-85.

Kusiak, A. 1990. *Intelligent Manufacturing Systems, 2nd Edition*, Prentice Hall International services in Industrial and Systems Engineering.

Lee-Post, A. 2000. Part family identification using a simple genetic algorithm. *International Journal of Production Research*, 38 (4): 793-810.

Malave C. and Ramachandran, S. 1994. Artificial Neural Networks for Intelligent Manufacturing. Department of Industrial Engineering, Texas A & M University, College Station. Chapman & Hall, London.

Mansouri, S. Moattat, H. and Newman, S. 2000. A Review of the Modern Approaches to Multi-Criteria Cell Design. *International Journal of Production Research*, 38 (5): 1201-1218.

Mukattash, A. 1997. *Automated Solution for Cell Formation*. Ph.D. Thesis, Department of Production Engineering and Metallurgy, University of Technology, Baghdad, Iraq.

Mukhopadhyay, S. Babu, K. and Sai, K. 2000. Modified Hamiltonian Chain: a Graph Theoretic Approach to Group Technology. *International Journal of Production Research*, 38 (11): 2459-2470.

Samhuri, M. 2000. *Genetic Algorithms for Robot Trajectory Planning*. Master Thesis, Industrial Engineering Department, University of Jordan, Amman, Jordan.

Sarker, B. and Mondal, S. 1999. Grouping Efficiency Measure in Cellular Manufacturing: a Survey and Critical Review. *International Journal of Production Research*, 37 (2): 285-314.

Smith, J. and Fogarty, T. 1997. Operator and Parameter Adaptation in Genetic Algorithms. *Soft Computing* 1: 81-87, Springer-Verlag.

Tiwari, M. K. and Vidyarthi, N. K. 2000. Solving Machine Loading Problems in a Flexible Manufacturing System Using a Genetic Algorithm Based Heuristic Approach. *International Journal of Production Research*, 38 (14): 3357-3384.

542413

Vajpayee, K. 1995. *Principles of Computer-Integrated Manufacturing, 1st Edition*. Prentice-Hall, Inc. A Simon & Schuster Company, Englewood Cliffs, New Jersey.

Winston, P. 1993. *Artificial Intelligence, 3rd Edition*. Addison-Wisely.

Zhao, L. Tsujimura, Y. and Gen, M. 1995. Genetic Algorithm for Fuzzy Clustering with Application to Manufacturing Cell Formation Problem. Intelligent Systems Engineering Laboratory, Department of Industrial and Systems Engineering, Ashikaga Institute of Technology, Ashikaga 326 Jaban.

Zolfaghari, S. and Liang, M. 1997. An Objective-Guided Ortho-Synapse Hopfield Network Approach to Machine Grouping Problems. *International Journal of Production Research*, 35 (10): 2773-2792.

APPENDIXES



MATLAB Software

Description

A.1 The Main Program: Graphical User Interface (GUI)

A comprehensive MATLAB functions are built and collected in a unitary software called “Genetic Algorithm Manufacturing Cell Formation Problem Solver” GAMCFPS. This software handle the whole process of grouping the machines/parts into cells/families.

The main characteristics of this software are the following:

- Hierarchical structure
- The flexibility to choose between two solving methodologies:
 1. Using the simple GA to solve the MCFP after specifying these parameters: population size, target standard deviation and mutation and crossover rates.
 2. Using metalevel GA where the control parameters (population size, target standard deviation and mutation and crossover rates) are determined by the first level GA and used to solve the second level GA, which gives the cell formation problem solutions.
- Ability to switch between two objective functions (grouping efficacy (GEC) ,and travelling salesman problem formulation (TSP)).
- Flexibility to specify the maximum number of cells (k_{max}) so that, the solved matrix may be partitioned into a number of cells less or equal to this number.
- Hybrid termination strategy (Target standard deviation, and maximum number of generations).

The Hierarchical structure provides the user the ability to modify or re-arrange the whole system by changing the functions and GUI's so as to modify the software performance.

The user can use the simple genetic algorithm to solve the MCFP by randomly tuning its parameters. Another new approach is to use metalevel GA where the GA is used to do

the two tasks: 1) optimize the GA parameters 2) solve the MCFP using these optimized parameters. The metalevel GA is a trial to find the GA parameters that will give the near optimal solution to all of the problems.

If the user does not want to get better near optimal solution, which is achieved by using metalevel GA, and he wants the results soon, the GAMCFPS software gives him the ability to specify certain GA parameters to see their results.

The main program MCFI.m; which stands for main cell formation interface; allows the user to enter the main menu (see Figure A.1) and perform the following functions (see appendix B for the different screens).

- 1- Choose between operating two levels of the metalevel GA or running the simple GA by setting its parameters.
- 2- On line help
- 3- Exit the program

The first level GA pushbutton activates metalevel.m function. This function performs the first task of the optimization methodology, which aims to produce the optimal genetic algorithm control parameters.

As mentioned in section 3.5 the control parameters of the first level GA are the standard GA control parameters specified by De Jong i.e., GAs (population size n , crossover rate P_c , mutation rate P_m), GAs (50, 0.6, 0.001).

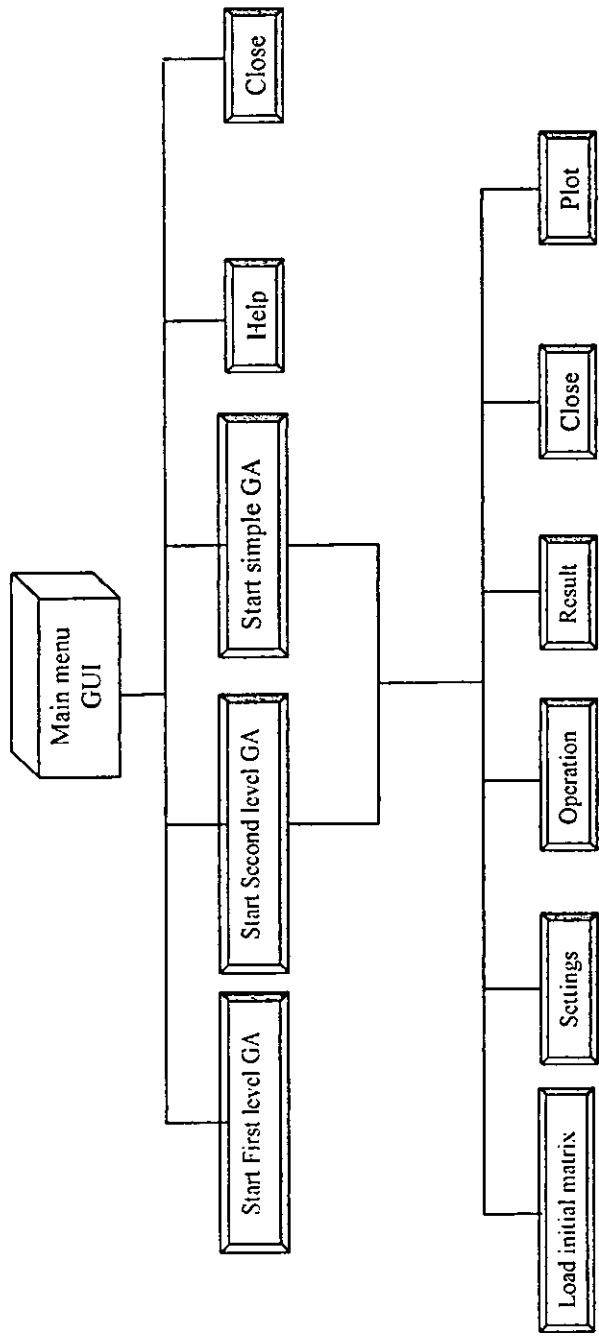


Figure A.1 The main menu of MATLAB software

The first level GA goes through several procedures, which are described in the flow diagram shown in Figure A.2:

1. Generation of 50 (population size) chromosomes. Each chromosome is coded as follows: chromosome 1: GA(n , TSD, P_c , P_m)
2. Execute the second level GA: the data set used here is data set number eight. It is selected because it has nearly the maximum number of exceptional elements and it has a large size relative to other data sets. The second level is executed using roulette wheel reproduction, grouping efficacy objective function, simple crossover, a combination of boundary and uniform mutation, elitism replacement strategy and hybrid termination criteria. The control parameters of the second level are taken from step one.
3. Evaluation of the first level GA. The first level GA is evaluated by assigning the best values produced in the second level GA to the fifty chromosome produced in step one.
4. After evaluating the chromosomes of the first level GA, these chromosomes are reproduced using roulette wheel, crossed over using simple crossover technique, and mutated using uniform mutation. The chromosomes which were crossed over and mutated are reevaluated using the second level GA. After that elitism replacement strategy and hybrid termination are used by the first level GA.
5. When the current number of generations reaches the specified number of generations, the GA system is stopped and the output is displayed on a special file which is constructed through the GA run. These results includes of course, the GA optimized control parameters.

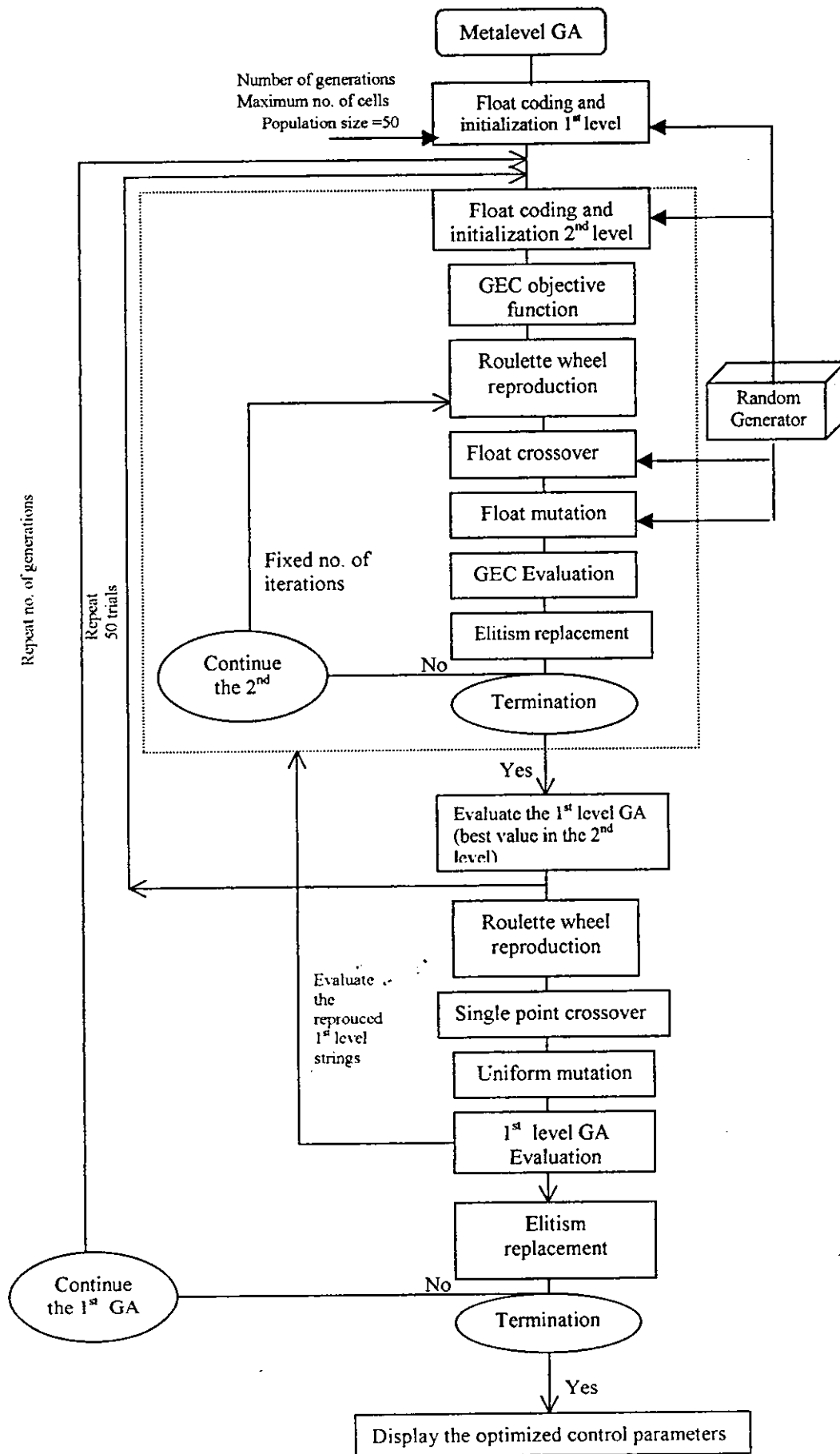


Figure A.2 The metalevel GA flow diagram

The second level GA pushbutton activates SLGAI.m which stands for second level GA interface. This GUI allows the user to do the following (see Figure A.3):

- Load the initial matrix by choosing it from a popup menu.
- specify the following settings:
 1. Genetic algorithm objective function.
 2. Maximum number of cells k_{max} .
 3. The optimized control parameters.
- Operate the second level GA to solve the cell formation problem.
- Display a plot of the best and average values of solutions variation with the number of generations.
- Display GAERI.m GUI, which stands for GA experimental results interface. This GUI displays the following:
 1. grouping efficacy
 2. grouping efficiency
 3. TSP results
 4. number of voids
 5. number of exceptional elements
 6. maximum number of cells
 7. The maximum number of generations

The start simple GA pushbutton activates SMGAI.m GUI, which stands for simple GA interface. This interface is identical to SLGAI.m except that it allows the user to specify tuned GA control parameters. So the user can examine the influence of any combination of the control parameters on the performance of the GA system.

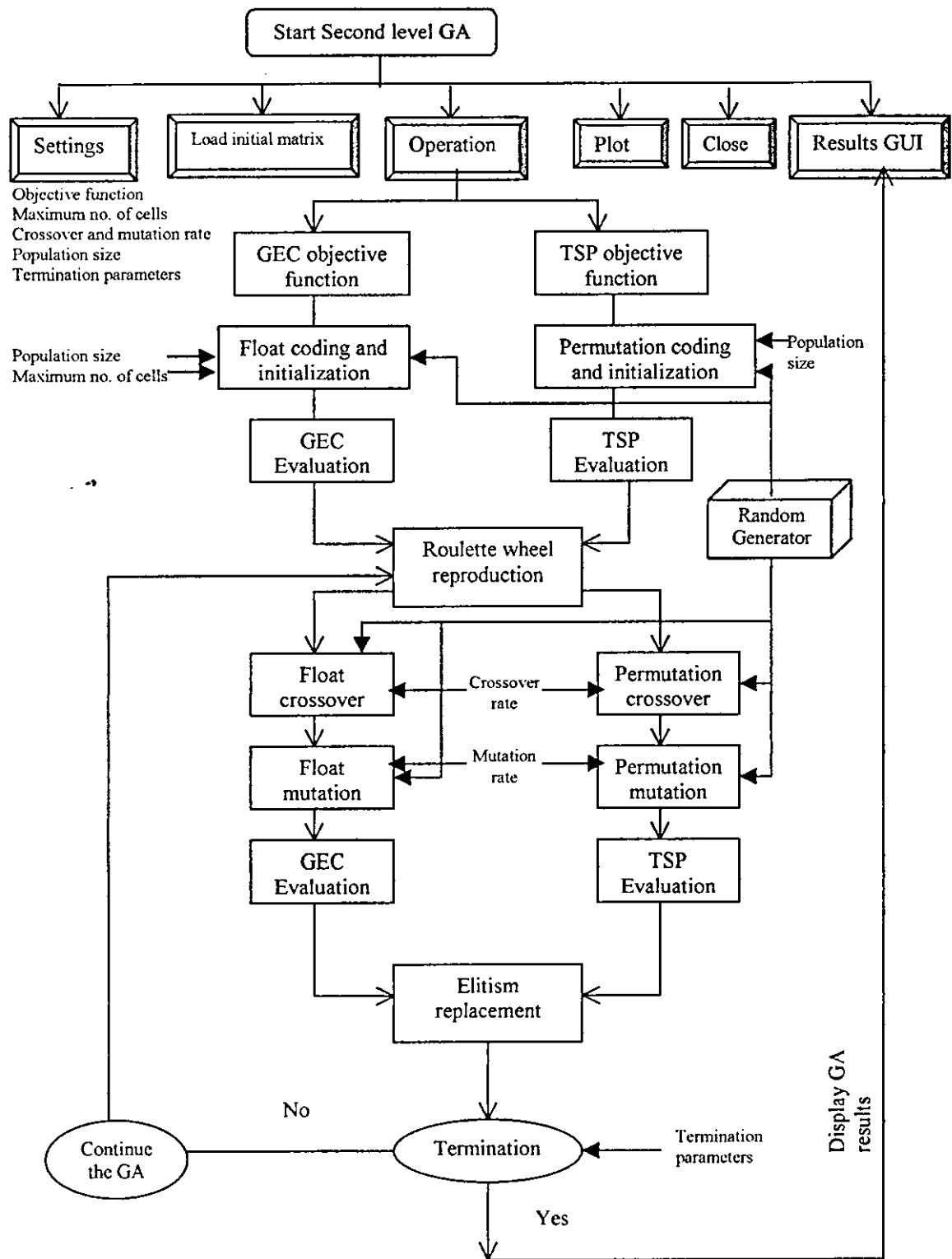


Figure A.3 The second level GA flow diagram

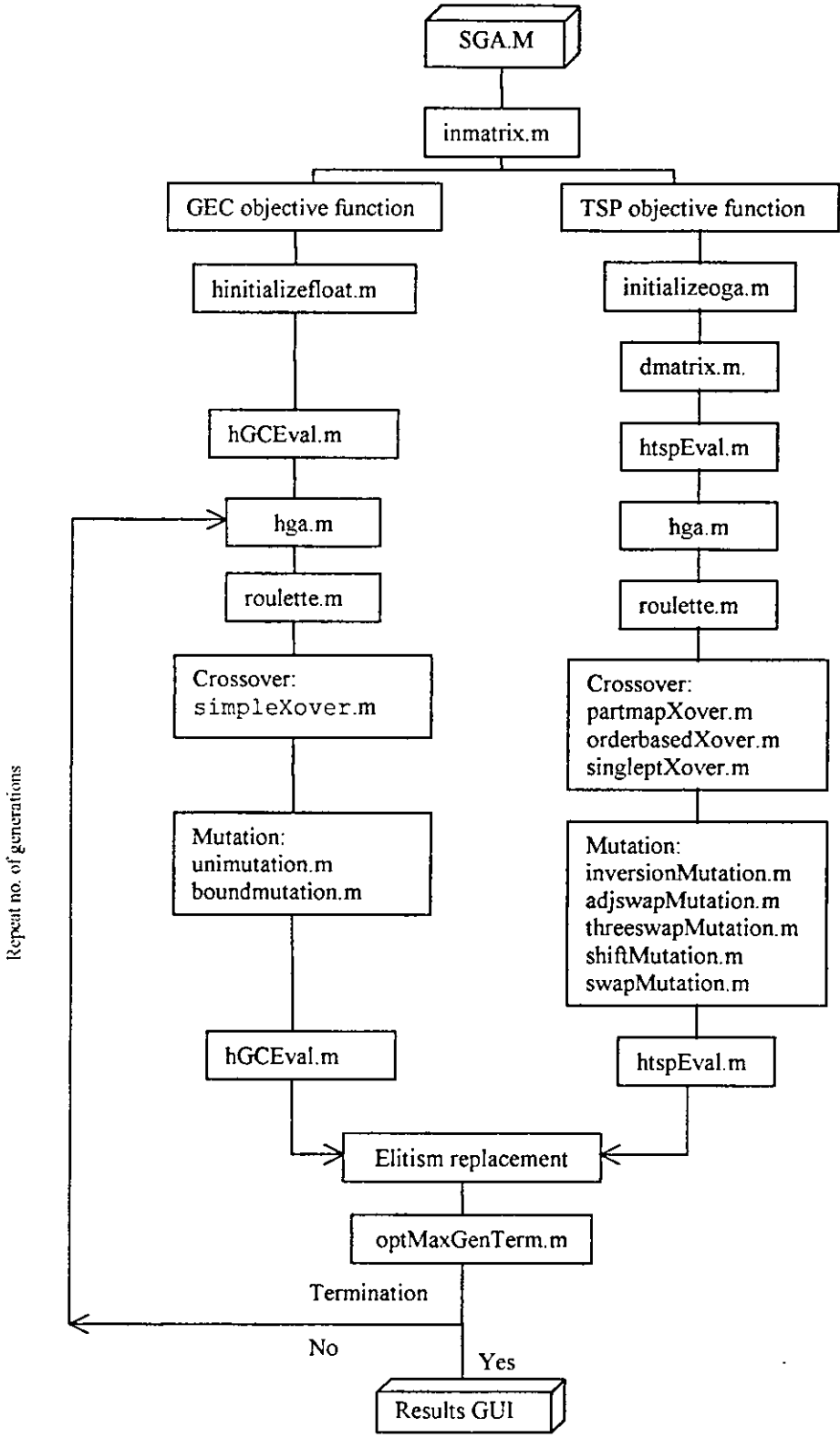


Figure A.4 The main functions of the MATLAB software.

2. Initialization function (`initializefloat.m`): This function generates randomly an initial population of individual strings for the GEC objective function, whereas, `initializeoga.m` for the TSP objective function.

Inputs:

- population size.
- Bounds on the values that are randomly chosen. These bounds are: number of machines and parts, and the maximum number of permissible cells.

Output:

- The randomly generated initial population.

3. Selection (reproduction) function (`roulette.m`): This function evaluates the population of solutions initially and after each generation. It uses roulette wheel method to copy strings according to their fitness value i.e. strings with a higher value have a higher probability of contributing one or more offspring in the next generation.

Input:

- the initial or intermediate population of solutions that are associated with their fitness values.

Output:

- the survived strings which will enter the mating pool.

4. Crossover functions: Crossover is performed according to the crossover rate (probability). There are three crossover techniques associated with the TSP objective function and one with the GEC, see Figure A.4.

Input:

- two randomly selected parents

Output:

- two crossed over offsprings

5. Mutation functions: mutation is performed according to the mutation rate (probability) designed by the first level GA. There are five mutation techniques associated with the TSP objective function and two with the GEC, see Figure A.4.

Input:

- two randomly selected parents

Output:

- two mutated offsprings

6. Evaluation function: During each generation chromosomes are evaluated using some measure of fitness. (tspEval.m for TSP and hGCEval.m for GEC)

Input:

- chromosomes with no value information carried with them.

Output:

- chromosomes with the fitness value attached to them.

7. Termination function (optMaxGenTerm.m): a hybrid termination criteria, which uses the target standard deviation in addition to the maximum number of generations .

Inputs:

- current generation number
- maximum number of generations

- current standard deviation
- target standard deviation

Outputs:

- done = 1 if one of the termination criteria is met
- done = 0 if the none of the termination criteria is met

8. Genetic algorithm function (hga.m): it is a high level function which contains the selection, crossover, mutation and evaluation functions, in addition to the elitism replacement method.

Input:

- the initial random population

Outputs:

- the final population and the near optimal solution.
- A trace of the best and average and standard deviation of GA solutions variation with the number of generations.

9. Second level GA solver (SGA.m): This is the highest level of all the 2nd level GA. It contains all the previously described functions.

Input:

- the initial data set matrix

Output:

- the solution matrix of the data set.



MATLAB Software

Screens

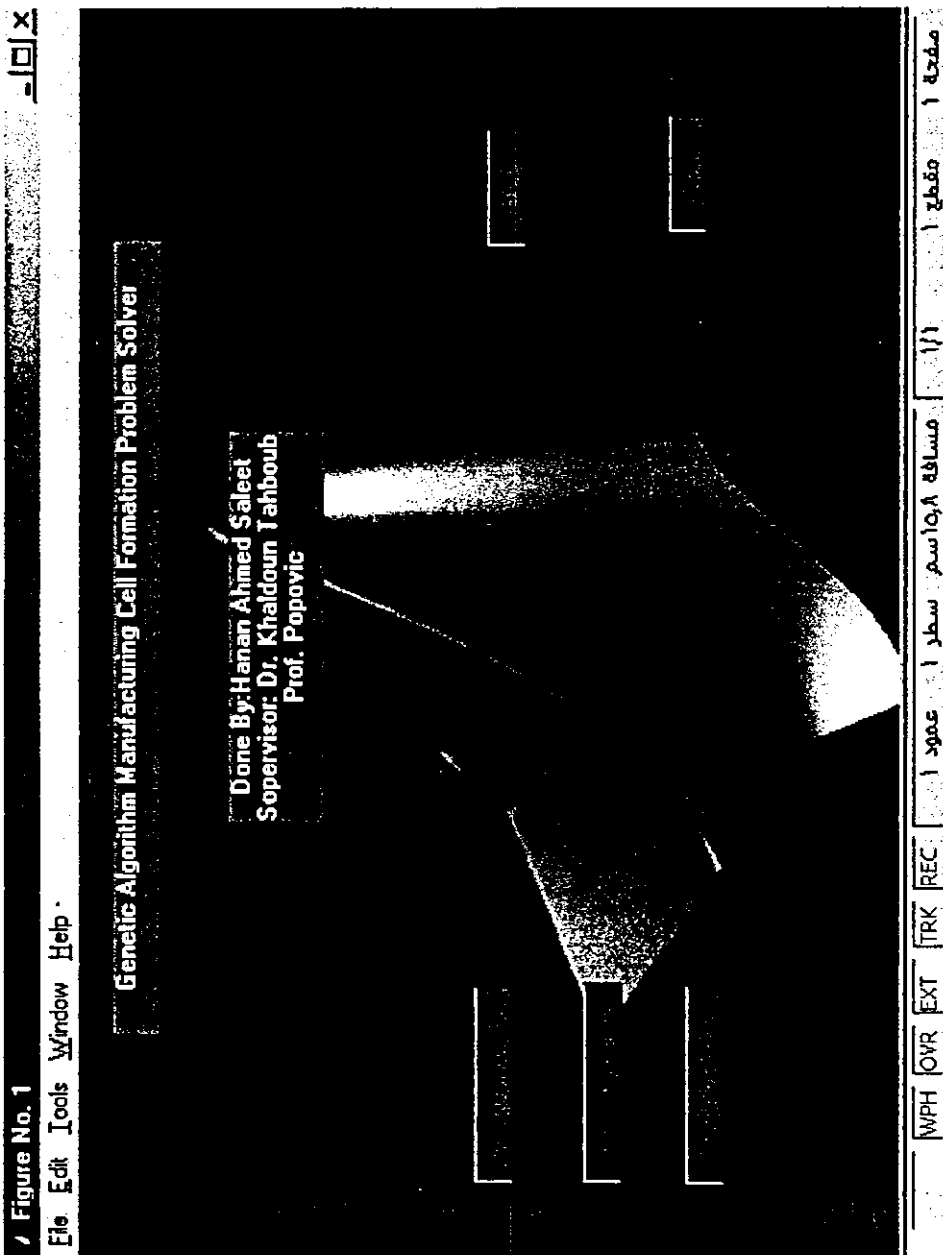


Figure No. 1

Figure B.1 The MATLAB software main menu

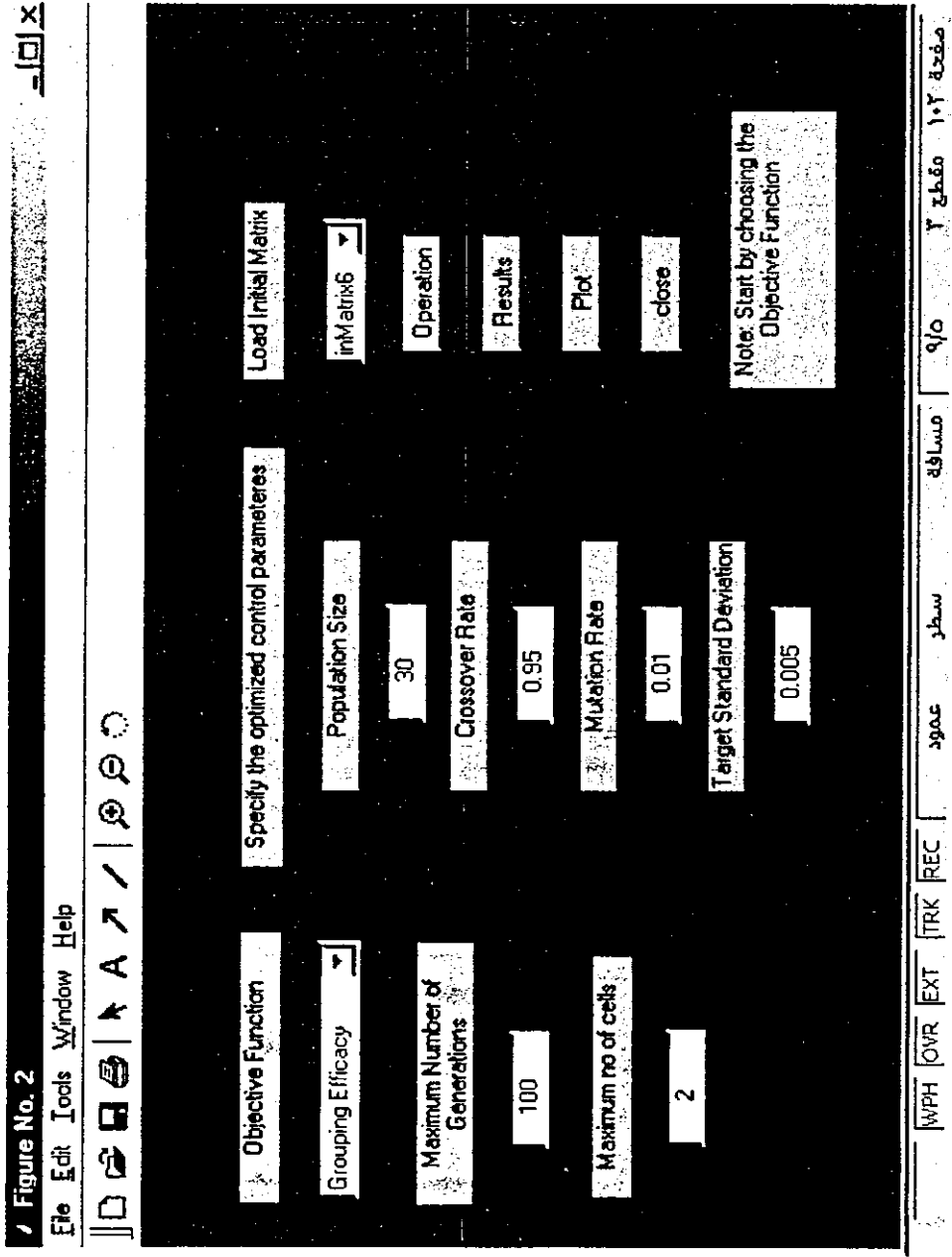


Figure B.2: The second level GA GUI

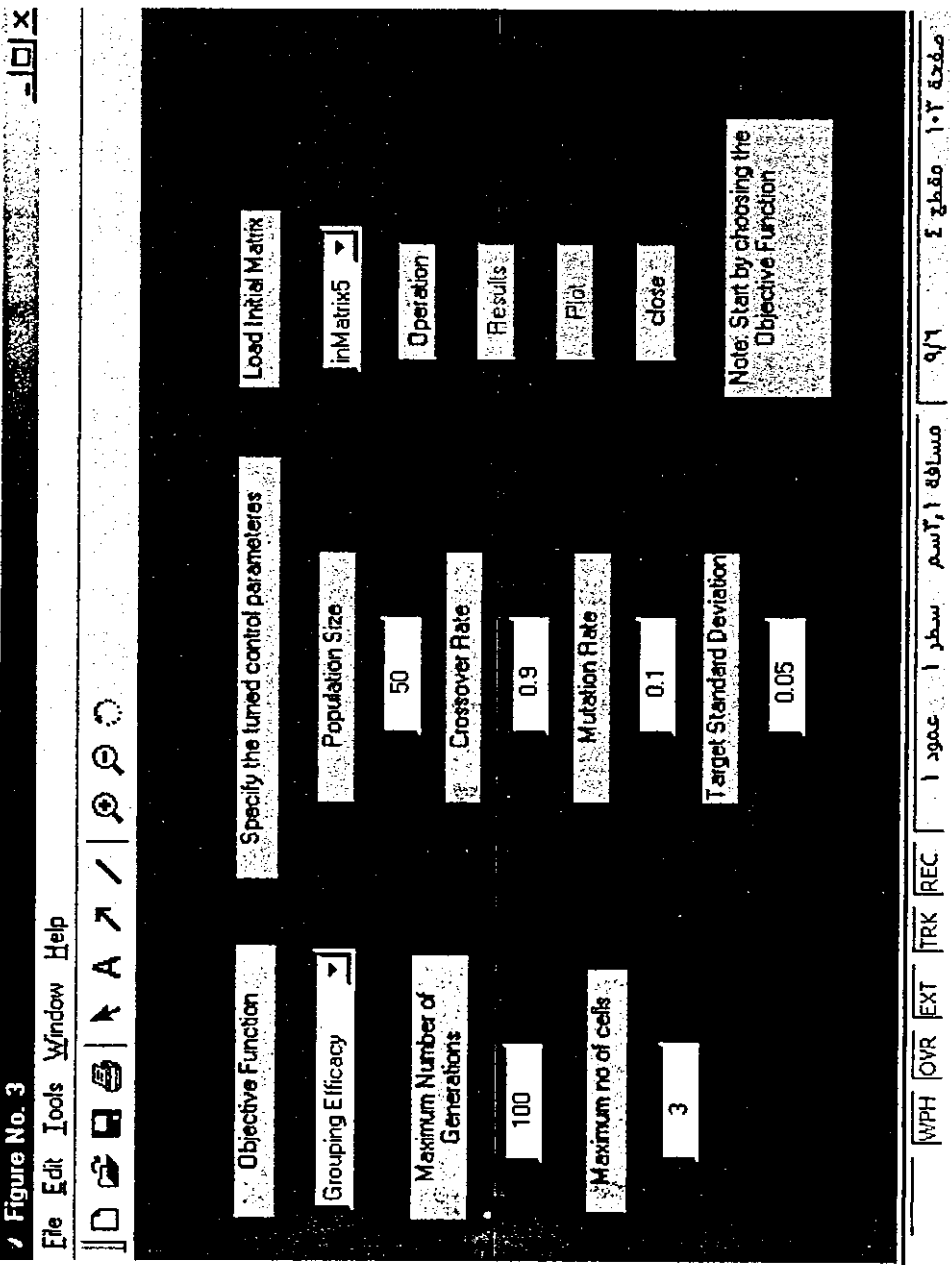


Figure B.3: The simple GA GUI

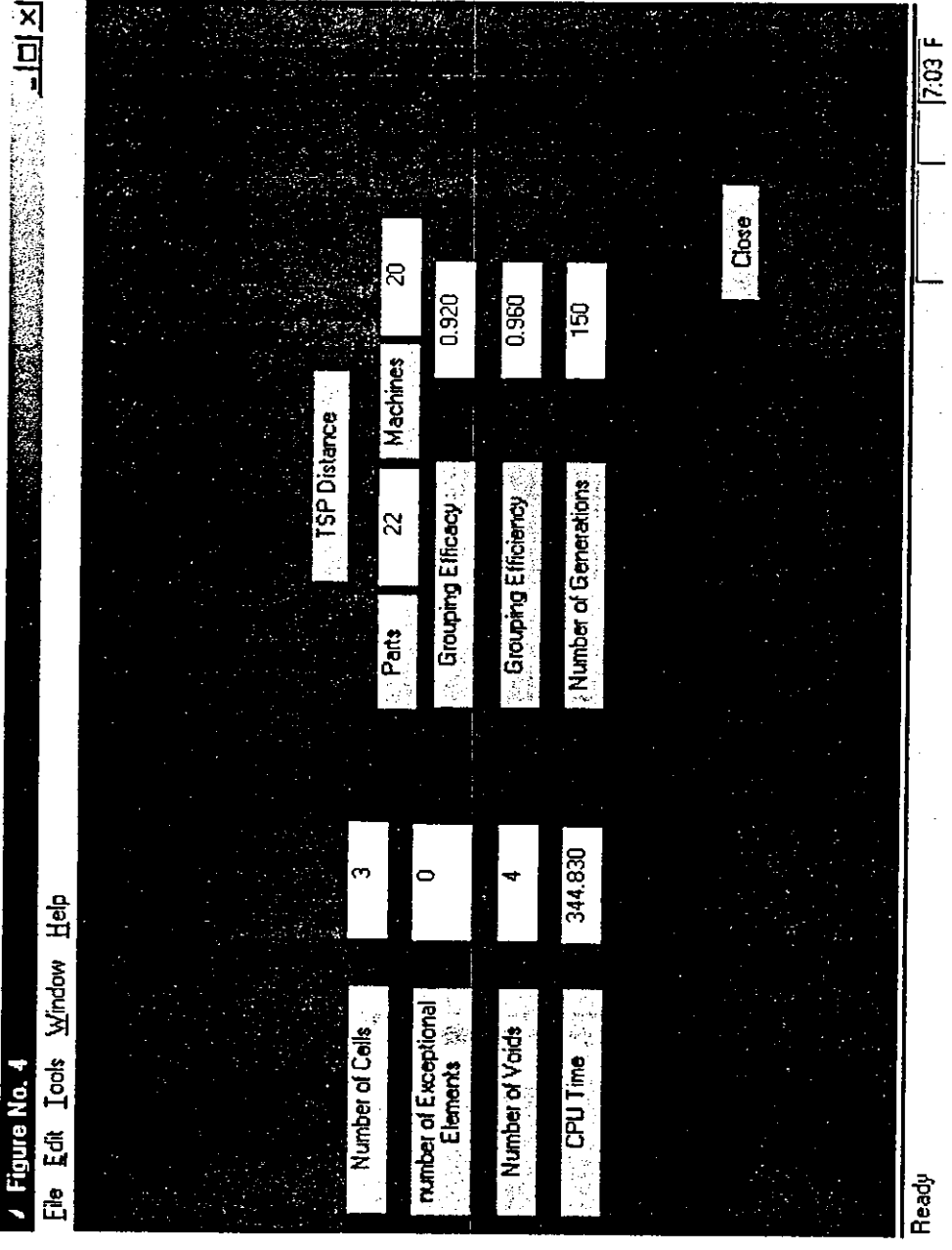


Figure B.4: The results GUI

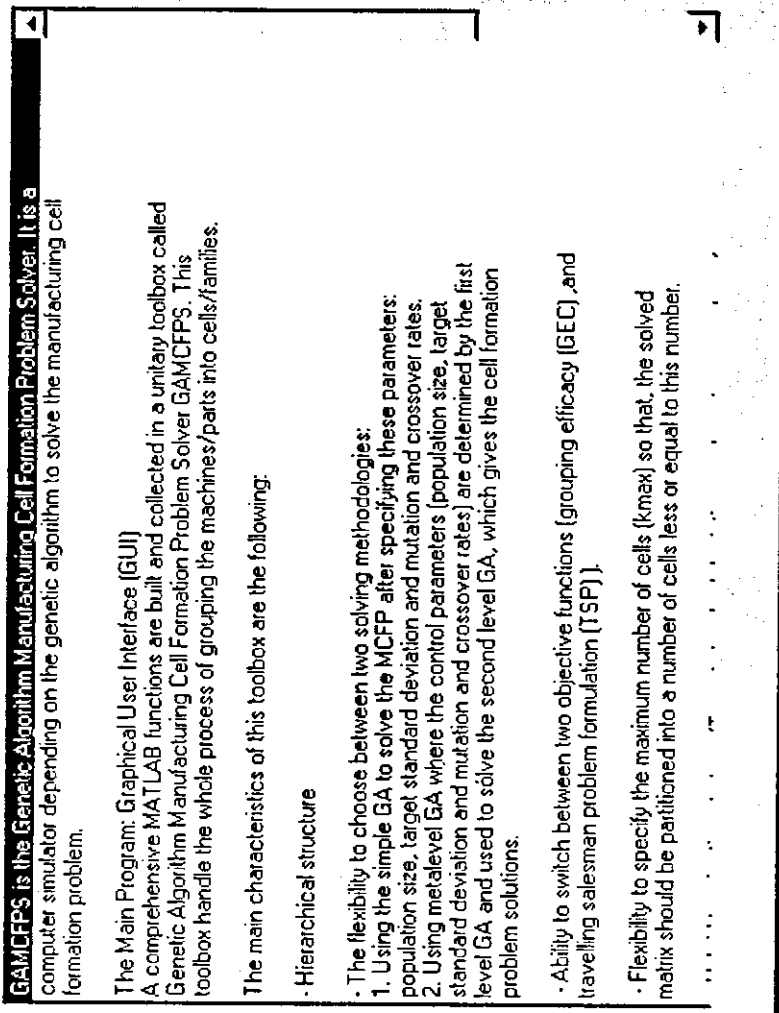


Figure B.5: the Help GUI

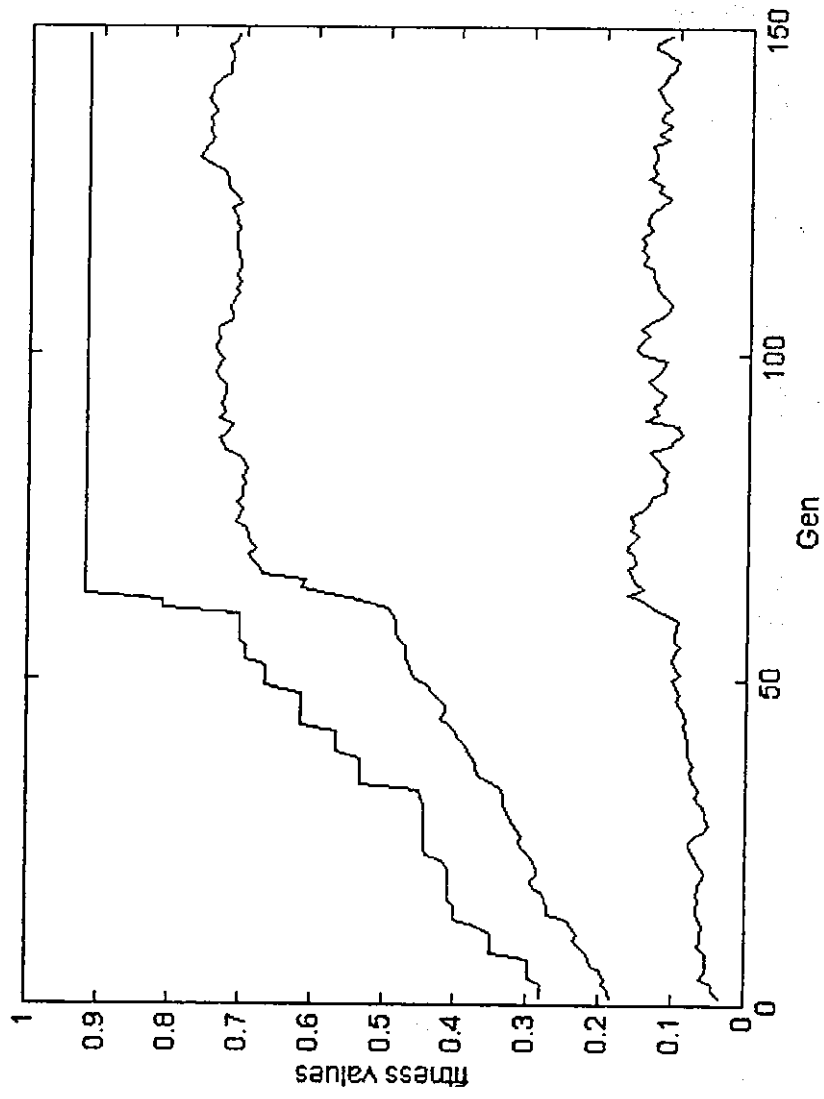


Figure B.6: Plot of the fitness values versus the number of generations.



Metalevel Genetic Algorithm
Solutions for the
Ten Data Sets

Table C.1a: The original incidence matrix for the 6 x 10 King (1980a) problem, data set number 1.

		Parts									
		1	2	3	4	5	6	7	8	9	10
Machines	1	0	0	0	0	1	0	0	1	0	0
	2	1	1	1	1	0	1	1	0	0	0
	3	1	1	1	1	0	1	0	0	0	0
	4	1	1	1	1	1	0	0	0	0	0
	5	0	0	0	0	1	0	0	0	1	0
	6	0	0	0	0	1	1	0	1	1	1

Table C.1b: The solution matrix for the 6 x 10 King (1980a) problem, data set 1.
 $\Gamma=72.4$, $\eta=86.3$, $CI=68.6$

		Parts									
		8	5	9	10	1	2	3	4	6	7
Machines	1	1	1	0	0	0	0	0	0	0	0
	5	0	1	1	0	0	0	0	0	0	0
	6	1	1	1	1	0	0	0	0	1	0
	2	0	0	0	0	1	1	1	1	1	1
	3	0	0	0	0	1	1	1	1	1	0
	4	0	1	0	0	1	1	1	1	0	0

Table C.2a: The original incidence matrix for the 6 x 15
Chen and Guerrero (1994) problem, data set number 2.

		Parts														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Machines	1	1	1	1	0	0	0	0	1	1	1	0	1	1	0	1
	2	0	1	1	1	0	1	0	0	1	1	0	0	1	0	1
	3	1	0	0	0	1	0	1	1	1	0	0	1	1	1	0
	4	1	1	0	0	0	0	0	1	0	0	1	0	1	0	1
	5	0	0	1	1	0	1	0	0	0	0	0	1	0	1	0
	6	0	0	0	1	0	1	1	0	0	0	0	0	0	1	0

Table C.2b: The near optimal solution matrix by GA for 6 x 15
Chen and Guerrero (1994) problem, data set 2.

$\Gamma=61.7, \eta=80.1, CI=61.8$

		Parts														
		4	6	14	5	7	12	1	2	3	8	9	10	11	13	15
Machines	5	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0
	6	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
	3	0	0	1	1	1	1	1	0	0	1	1	0	0	1	0
	1	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1
	2	1	1	0	0	0	0	0	1	1	0	1	1	0	1	1
	4	0	0	0	0	0	0	1	1	0	1	0	0	1	1	1

Table C.2c: The optimal solution matrix for 6 x 15 Chen and
Guerrero (1994) problem, data set 2.

$\Gamma=64.2, CI=63.7$

		Parts														
		5	4	6	7	12	14	1	2	3	8	9	10	11	13	15
Machines	3	1	0	0	1	1	1	1	0	0	1	1	0	0	1	0
	5	1	1	1	0	1	1	0	0	1	0	0	0	0	0	0
	6	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	1	0	1	1	1	1	1	1	0	1	1
	2	0	1	1	0	0	0	0	1	1	0	1	1	0	1	1
	4	0	0	0	0	0	0	1	1	0	1	0	0	1	1	1

Table C.3a: The original incidence matrix for the 9 x 9 Chu and Hayya (1991) problem, data set number 3.

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

Table C.3b: The solution matrix for the 9 x 9 Chu and Hayya (1991) problem, data set number 3.

$\Gamma=74.3$, $\eta=89.1$, $CI=70.4$

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

Table C.4a: The original incidence matrix for the 10 x 8 King (1980b) problem, data set number 4.

		Parts							
		1	2	3	4	5	6	7	8
Machines	1	1	0	1	0	0	1	0	1
	2	1	0	0	0	0	1	0	0
	3	0	1	0	0	1	0	0	1
	4	1	0	1	0	0	1	0	0
	5	0	0	0	1	0	0	1	0
	6	0	1	0	0	1	0	0	1
	7	0	0	0	0	1	0	0	1
	8	1	0	1	0	0	1	0	0
	9	0	0	0	1	0	0	1	0
	10	0	1	0	0	0	0	1	0

Table C.4b: The solution matrix for the 10 x 8 King (1980b) problem, data set number 4.

$\Gamma=82.2, \eta=92.5, CI=81.9$

		Parts							
		1	2	3	4	5	6	7	8
Machines	2	1	0	1	0	0	0	0	1
	1	1	1	1	0	0	0	0	0
	8	1	1	1	0	0	0	0	0
	4	1	1	1	0	0	0	0	0
	9	0	0	0	1	1	0	0	0
	5	0	0	0	1	1	0	0	0
	10	0	0	0	0	1	1	0	0
	3	0	0	0	0	0	1	1	1
	6	0	0	0	0	0	1	1	1
	7	0	0	0	0	0	0	1	1

Table C.5a: The original incidence matrix for the 10 x 12 Viswanathan (1996) problem, data set number 5.

		Parts											
		1	2	3	4	5	6	7	8	9	10	11	12
Machines	1	0	0	0	0	0	1	1	0	1	0	0	0
	2	0	0	0	1	0	1	0	0	0	0	0	0
	3	1	1	1	1	0	0	0	0	0	0	0	0
	4	1	1	0	0	0	0	0	0	0	0	0	0
	5	0	1	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	1	1	1	1	1	0	0	1	1
	7	0	0	0	1	0	1	0	1	0	1	1	0
	8	0	0	0	0	1	1	1	1	0	0	0	0
	9	0	0	0	1	0	0	1	1	1	1	1	1
	10	0	1	0	1	1	0	0	1	0	1	0	1

Table C.5b: The solution matrix for the 10 x 12 Viswanathan (1996) problem, data set number 5.
 $\Gamma=59.6$, $\eta=80.5$, $CI=58.3$

		Parts											
		1	2	3	6	7	9	4	5	8	10	11	12
Machines	3	1	1	1	0	0	0	1	0	0	0	0	0
	4	1	1	0	0	0	0	0	0	0	0	0	0
	5	0	1	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	1	1	1	0	0	0	0	0	0
	2	0	0	0	1	0	0	1	0	0	0	0	0
	8	0	0	0	1	1	0	0	1	1	0	0	0
	6	0	0	0	1	1	0	1	1	1	0	1	1
	7	0	0	0	1	0	0	1	0	1	1	1	0
	9	0	0	0	0	1	1	1	0	1	1	1	1
	10	0	1	0	0	0	0	1	1	1	1	0	1

Table C.6a: The original incidence matrix for 15 x 10 simple Chan and Milner problem (1982), data set number 6.

		Parts									
		1	2	3	4	5	6	7	8	9	10
Machines	1	0	0	1	1	0	1	0	0	0	0
	2	1	0	0	0	0	0	1	0	0	1
	3	0	1	0	0	1	0	0	1	0	0
	4	0	0	0	1	0	1	0	0	1	0
	5	0	1	0	0	1	0	0	1	0	0
	6	0	0	1	0	0	1	0	0	1	0
	7	0	0	0	0	0	0	1	0	0	1
	8	0	1	0	0	1	0	0	1	0	0
	9	0	0	1	1	0	1	0	0	1	0
	10	1	0	0	0	0	0	1	0	0	1
	11	1	0	0	0	0	0	1	0	0	1
	12	1	0	0	0	0	0	1	0	0	1
	13	0	1	0	0	1	0	0	1	0	0
	14	0	0	1	1	0	1	0	0	1	0
	15	1	0	0	0	0	0	1	0	0	1

Table C.6b: Metalevel GA solution for the 15 x10 Simple Chan and Milner problem (1982), data set number 6.

$\Gamma=92.0$, $\eta=96.0$, $CI=92.8$

		Parts									
		5	2	8	1	7	10	3	9	6	4
Machines	13	1	1	1	0	0	0	0	0	0	0
	3	1	1	1	0	0	0	0	0	0	0
	8	1	1	1	0	0	0	0	0	0	0
	5	1	1	1	0	0	0	0	0	0	0
	7	0	0	0	1	1	1	0	0	0	0
	2	0	0	0	0	1	1	0	0	0	0
	10	0	0	0	1	1	1	0	0	0	0
	12	0	0	0	1	1	1	0	0	0	0
	15	0	0	0	1	1	1	0	0	0	0
	11	0	0	0	1	1	1	0	0	0	0
	1	0	0	0	0	0	0	1	1	1	0
	4	0	0	0	0	0	0	0	1	1	1
	14	0	0	0	0	0	0	1	1	1	1
	9	0	0	0	0	0	0	1	1	1	1
	6	0	0	0	0	0	0	1	0	1	1

Table C.7a: The original incidence matrix for the 15 x 10 Complex Chan and Milner problem (1982), data set number 7.

		Parts									
		1	2	3	4	5	6	7	8	9	10
Machines	1	0	0	1	1	0	0	0	1	1	0
	2	1	0	0	0	0	1	1	1	0	0
	3	0	0	1	1	0	0	0	1	1	0
	4	0	0	0	1	0	0	0	1	1	0
	5	0	0	0	0	1	0	0	0	0	1
	6	1	0	1	1	0	1	1	1	1	0
	7	1	0	0	0	0	1	0	0	1	0
	8	0	1	0	0	1	0	0	0	0	1
	9	0	1	0	0	1	0	0	0	0	1
	10	0	1	0	0	1	0	0	0	0	1
	11	1	0	0	0	0	1	1	0	0	0
	12	1	0	0	0	0	1	1	0	0	0
	13	0	1	0	0	1	0	0	0	0	1
	14	0	0	1	1	0	0	0	1	1	0
	15	0	1	0	0	1	0	0	0	0	1

Table C.7b: Metalevel GA solution for the 15 x10 Complex Chan and Milner problem (1982), data set number 7.

$\Gamma=85.4, \eta=94.5, CI=86.1$

		Parts									
		3	4	8	9	2	5	10	1	6	7
Machines	14	1	1	1	1	0	0	0	0	0	0
	3	1	1	1	1	0	0	0	0	0	0
	4	0	1	1	1	0	0	0	0	0	0
	1	1	1	1	1	0	0	0	0	0	0
	6	1	1	1	1	0	0	0	1	1	1
	15	0	0	0	0	1	1	1	0	0	0
	13	0	0	0	0	1	1	1	0	0	0
	10	0	0	0	0	1	1	1	0	0	0
	9	0	0	0	0	1	1	1	0	0	0
	8	0	0	0	0	1	1	1	0	0	0
	5	0	0	0	0	0	1	1	0	0	0
	12	0	0	0	0	0	0	0	1	1	1
	11	0	0	0	0	0	0	0	1	1	1
	2	0	0	1	0	0	0	0	1	1	1
	7	0	0	0	1	0	0	0	1	1	0

Table C.8a: The original incidence matrix for the 8 x 20 Chandrasekharan and Rajagopalan (1986) problem, data set number 8.

	Parts																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20				
Machines	1	0	1	1	0	0	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	0
	2	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
	3	0	1	0	0	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0	1	0	1
	4	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1
	5	1	0	0	0	1	1	0	0	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0
	6	1	0	0	0	1	0	0	0	1	1	0	1	0	0	1	0	0	1	0	0	0	0	1
	7	0	0	1	1	0	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	1	0
	8	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0

Table C.8b: The solution matrix for for the 8 x 20 Chandrasekharan and Rajagopalan (1986) problem, data set number 8.

$$\Gamma=85.2, \eta=95.8, CI=84.4$$

	Parts																			
	3	4	6	7	18	20	1	5	10	12	15	2	8	9	11	13	14	16	17	19
Machines	2	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	4	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	7	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1
	8	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0
	6	0	0	0	0	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0
	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Table C.9a: The original incidence matrix for 14 x 24 King (1980) problem, data set number 9.

		Parts																							
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Machines	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1
	5	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0
	6	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	1	0
	7	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	1	0	0	0	0	1	1	0	1	0	1	1	0	0	0	0	0	0	1	0
	9	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0
	10	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	11	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	12	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	13	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
	14	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0

Table C.9b the first solution matrix for the 14 x 24 King (1980b)problem, data set 9.
 $\Gamma=61.4, \eta=81.3, CI=61.2$

		Parts																							
		3	4	21	24	6	7	8	18	1	2	17	19	20	23	5	9	10	11	12	13	14	15	16	22
Machines	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	13	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1
	8	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	0	1	0	1	1	1
	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	1	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	1	0

Table C.9c the second solution matrix for the 14 x 24 King (1980b) problem, data set 9.
 $\Gamma=61.8, \eta=81.4, CI=62.4$

	Parts																							
	3	4	21	24	6	7	8	18	1	2	17	19	20	23	5	9	10	11	12	13	14	15	16	22
2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	0	1	0	1	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	1	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0

Table C.10a: The original incidence matrix for 20 x 35 Burbidge problem (1960), data set number 10.

		Parts																																							
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35					
Machines	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0				
	2	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0			
	3	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0			
	4	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0			
	5	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		
	6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0		
	7	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	8	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0		
	9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0		
	10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0		
	11	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	12	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
	13	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	14	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	17	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	18	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	19	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table C.10b: Metalevel GA solution for the 20 x 35 Burbidge problem (1960), data set number 10.
 $\Gamma=75.8$ $\eta=88.1$ $CI=75.5$

		Parts																																						
		1	3	5	15	17	20	23	25	29	4	6	9	11	21	28	30	32	33	35	2	7	10	12	13	18	24	27	31	8	14	16	19	22	26	34				
Machines	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	11	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
	12	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	15	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	16	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	19	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ملخص إنشاء خلايا التصنيع المرنة باستخدام الخوارزميات الجينية

إعداد
حنان أحمد سليط

المشرف
د. خلدون طهبوب

المشرف المشارك
أستاذ دكتور دوبريفوجيه بوبوفيتش

تعد مشكلة إنشاء خلايا التصنيع المرنة الحجر الأساس في تطبيق تكنولوجيا المجموعات (Group Technology) لمواكبة الطرق الحديثة في التصنيع ، إذ يتم تجميع الآلات والمنتجات ضمن خلايا منفصلة للتقليل من عدد المنتجات التي يحتاج تصنيعها الى الانتقال من خلية لأخرى داخل المصنع.

يقوم هذا البحث باستخدام الخوارزميات الجينية (Genetic Algorithms) لحل مشكلة إنشاء الخلايا الصناعية المرنة حيث يتم حل المشكلة على مستويين : الأول، تقوم الخوارزميات الجينية باستخراج العوامل الفضلى (Optimal Parameters) التي تناسب جميع المشاكل المتوفرة. الثاني، تقوم الخوارزميات الجينية باستخدام هذه العوامل لاستخراج الحلول والنتائج.

لقد بينت عمليات المحاكاة باستخدام الحاسوب (Simulation) أن نتائج هذه الدراسة تماثل النتائج التي توصلت إليها الأبحاث الحديثة ، وتتميز بما يلي: ١- استخدام اقترانين مختلفين لحل الخوارزميات الجينية يمكن الباحث من الحصول على نتائج ذات خيارات متعددة. ٢- إمكانية وضع قيود (Constraints) على عدد الخلايا التي يجب أن يحتويها الحل النهائي. ٣- بناء برمجية للقيام بعمليات المراقبة و التحكم لجميع مراحل إنشاء الخلايا الصناعية.