An Integrated Approach to the Design of Cellular Manufacturing Systems for Dynamic Production Requirements

Fantahun Melaku Defersha

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montréal, Québec, Canada

February 2006

Your file Votre référence
ISBN: 978-0-494-16296-5
Our file Notre référence
ISBN: 978-0-494-16296-5

ABSTRACT

An Integrated Approach to the Design of Cellular Manufacturing Systems for Dynamic Production Requirements

Fantahun Melaku Defersha, Ph.D.

Concordia University, 2006

Due to increasing international competition, shorter product life-cycles, variable demand, diverse customer needs and customized products, manufacturers are forced from mass production to the production of large product mixes. In order to adapt to such changes, firms are required to make their manufacturing systems efficient and flexible. Traditional manufacturing systems, such as job shops and flow lines, cannot provide the required efficiently coupled with flexibility to handle these changes. Cellular manufacturing (CM), which incorporates the flexibility of job shops and the high production rate of flow lines, has emerged as a promising alternative. Although CM provides great benefits, its design process is complex for real-life problems. The design process should pass through a number of steps involving several structural and operational aspects. The first important critical step is the formation of part families and machine cells. The effectiveness of this design step heavily depends on the proper consideration of several relevant factors. To this end, a model that incorporates various pragmatic issues is essential. This research is aimed at the development of comprehensive mathematical models to serve in the design of cellular manufacturing systems for dynamic production requirements. In this work, two different mathematical models have been proposed and efficient solution

procedures are developed to solve these models.

The first mathematical model addresses the design of a dynamic cellular manufacturing system. In this model, the product mix is assumed to vary from period to period where the production quantity of each product during each period is a given data. System reconfiguration is considered to respond to the changing product mix variation. In addition to dynamic system reconfiguration, the model incorporates several pragmatic issues such as alternative routings, lot splitting, sequence of operations, multiple units of identical machines, machine capacity, workload balancing among cells, operation cost, cost of subcontracting part processing, tool consumption cost, setup cost and other practical constraints. The solution of this model is obviously NP-hard. Off-shelf optimization software cannot be used to solve real size problems of this nature. In order to solve real size problems efficiently, we develop two different heuristics: one based on genetic algorithm and the other based on simulated annealing. The algorithms have been implemented both on a sequential and a parallel computing environments. Numerical examples show that the proposed methods are efficient in solving the proposed model for large size problems.

The second mathematical model addresses an integrated approach to the design of dynamic cellular manufacturing systems and production planning in MRP environment. The major difference of second model from first one is that in the second model the production lot size of each product during each period is a decision variable but not a given data. The model also consider the effect of lot sizes on the product quality. Proponents of the Just-In-Time philosophy contend that smaller lot sizes result in improved product quality. Others (from the disruptive philosophy) argue that smaller lot sizes result in excessive interruptions and disruptive environments that impair learning and increase defects. The cell formation model is formulated to account for either philosophy allowing the model user to

select his/her preference. In order to solve this integrated cell formation and production planning model, two search heuristics, one based on genetic algorithm and the other based on simulated annealing, have been developed. During the course of the search, these methods interactively use the simplex algorithm in ILOG CPLEX to solve a Linear Programming sub-problem corresponding to each solution visited. Numerical examples showed that the proposed method were efficient. The obtained solution is finally submitted, as a starting point, to a post-optimization module. The post-optimization module is the general branch and cut algorithm in ILOG CPLEX attempting to solve the proposed MIP model starting from a known good solution.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to **Dr. Mingyuan Chen** of Concordia University for his time, support and guidance through out the course of this research. He spent many hours of his time to teach and guide me. His supervision and encouragement was really of a paramount importance to me in completing this work. The journal papers and refereed conference proceedings published from my thesis and those submitted for review were made possible by his most responsive follow-ups and incisive critiques on the original drafts. It was really my great pleasure to have worked with him. I would like to acknowledge the examiners Dr. Ming Liang, Dr. Dennis Kira, Dr. Mojtaba Kahrizi, Dr. Ali Akgunduz, Dr. Chevy Chen, and Dr. Kudret Demirli for their time and helpful suggestions. I would also thank Dr. Dong Cao of Quebec University (Trois-Rivieres) for the conference paper we authored which helped me in developing ideas and methods in some of the chapters of this thesis.

My gratitude goes to **Concordia University** for awarding me four different fellowships and scholarships. These fellowships and scholarships are (1) Concordia University Graduate Fellowship, (2) Concordia University International Tuition Fees Remission Award, (3) Campaign for a New Millennium Graduate Scholarship - School of Graduate Studies, and (4) Concordia University Partial Tuition for International Students. I also would like to express my appreciations for the funding from Concordia ENCS SRT Grant and NSERC Research Grant through Dr. Mingyuan Chen.

My gratitude also goes to **RQCHP** (Réseau québécois en calcul de haute performance) for granting me access to the parallel computing facilities in University

of Montreal and University of Sherbrooke. The parallel computational experiments reported in this work were made possible using the system available in University of Sherbrooke.

I would like to thank my wife, **Meseret Amare Hailu**, for her enduring support, comfort, and understanding over the years so that I could pursue what she felt was important to me. Though she is in a difficult time for her life, she has been giving all her love and sacrifice to support me. She has shared the journey of PhD thesis writing from its early beginnings to the last stage and I acknowledge with gratitude and love the importance of her presence in my life. I would have not achieved this goal without her support. Also, I thank my children, **Hildana** and **Ephrem**, for bringing a lot of fun and pleasure into my life and becoming a good reason to work hard in my graduate study.

But above all, I thank my Lord and Savior **JESUS CHRIST** for granting me eternal life in heaven through His death and resurrection. I thank Him for His all-sufficient grace, so freely available to me and to everybody. I thank Him for the wisdom He gave me to deal with the challenges I faced while doing my research. I thank Him for the strength and patience He gave me to accomplish this thesis work. Jesus said to him (to Thomas) "*I am the way, and the truth, and the life. No one comes to the Father except through me.*" John 14:6.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

t	-	Time period index: $t = 1, 2, ..., T$
i	-	Part type index: $i = 1, 2, ..., I$
r	-	Route index of part i: $r = 1, 2, \cdots, R_i$
j	-	Index of operations of part i: $j = 1, 2, ..., J_i$
k	-	Machine index: $k = 1, 2, ..., K$
a	-	Tool index: $a = 1, 2, ..., G$
l	-	Cell index: $l = 1, 2, ..., L$
$d_i(t)$	-	Demand for part i in time period t
V_i	-	Unit cost to move part i between cells
B_i	-	Batch size of product i
Φ_i	-	Cost of subcontracting part i
λ_{jig}	-	Equals to 1 if operation j of part i requires tool g; 0, otherwise
δ_{gk}	-	Equals to 1 if tool g is available on machine k; 0, otherwise
h_{jik}	-	Processing time of operation j of part i on machine type k
w_{jik}	-	Tool consumption cost of operation j of each part i on machine type k
μ_{jik}	-	Setup cost for operation j of part i on machine type k
$Q_k(t)$	-	Maximum number of machine type k that can be procured at the beginning of period t
$P_k(t)$	-	Procurement cost of machine type k at the beginning of period t
G_k	-	Maintenance and overhead costs of machine type k per time period t
O_k	-	Operation cost per hour of machine type k
C_k	-	Capacity of one machine of type k for one time period

LB_l	-	Minimum number of machines in cell l
UB_l	-	Maximum number of machines in cell l
R_k^+	-	Cost of installing one machine of type k
R_k^-	-	Cost of removing one machine of type k
q	-	$0 \leq q < 1$; A factor for the work load of a cell being as low as $q \times 100\%$ from the average work load per cell
$Z_i(t)$	-	The number of cells among which an entire lot of part i may split into during time period t for the processing of certain operations; $Z_i(t) \in \{1,\ 2, ...,\ L\}$
Θ	-	A set of machine pairs $\{(k^a, k^b)/k^a, k^b \in \{1, \cdots, K\}, k^a \neq k^b,$ and k^a cannot be placed in the same cell with $k^b\}$
Ω	-	A set of machine pairs $\{(k^c, k^d)/k^c, k^d \in \{1, \cdots, K\}, k^c \neq k^d,$ and k^c should be placed in the same cell with $k^d\}$
$N_{kl}(t)$	-	Number of type k machines to assign to cell l at the beginning of period t
$y_{kl}^+(t)$	-	Number of type k machines to add to cell l at the beginning of period t
$y_{kl}^-(t)$	-	Number of type k machines to remove from cell l at the beginning of period t
$\eta_{jikl}(t)$	-	The proportion of the total demand of part i with the j^{th} operation to perform by machine type k in cell l during period t
$\bar{\eta}_i(t)$	-	The proportion of the total demand of part i to be subcontracted in time period t
$r_{kl}(t)$	-	Equals to 1, if type k machines are to be assigned to cell l during time period t; 0, otherwise
$p_{jil}(t)$	-	Equals to 1, if operation j of part i is to be processed in cell l

		during period t; 0, otherwise
Θ'	-	Set of machines (unpaired) included in Θ
ϑ, φ	-	Counter variables,
$\Theta_{\vartheta\varphi}$	-	Set of machines from which φ^{th} machine of the ϑ^{th} set is to be arbitrarily taken,
$k_{\vartheta\varphi}$	-	The φ^{th} machine of the ϑ^{th} set arbitrarily taken from $\Theta_{\vartheta\varphi}$,
Δ	-	A function that take a set of machines $\Theta_{\vartheta\varphi} \setminus \{k_{\vartheta\varphi}\}$ and machine $k_{\vartheta q}$ as the first and second arguments respectively and returns a set of machine $\Theta_{\vartheta,\ \varphi+1}$, taken from its first argument, that can be placed in the same cell with machine type $k_{\vartheta\varphi}$,
Υ_ϑ	-	The ϑ^{th} set of machines taken from Θ' that can be placed in the same cell
g	-	Generation counter, $g = 1, 2, ..., g_{max}$
g_{max}	-	Maximum number of generations
$PP(g)$	-	Parents population of generation g
$CP(g)$	-	The current children population which makes up $PP(g+1)$
E	-	Fitness (Energy) function
BI_p	-	Best feasible individual so far found in the p^{th} processor
BI	-	Best feasible individual so far found (by all the processors)
$Phase$	-	The current phase of the search which equals to 1 for the first phase or 2 for the second phase
g_{phase}	-	Generation at which the value of $Phase$ should be set equal to 2 if it were not previously set to this value by other conditions,
w	-	Number of successive generations counted without any improvement of the best individual so far found
w_{max}	-	Maximum value of w at which point population rejuvenation is to be performed

b	-	Number of successive population rejuvenations counted without any improvement of the best individual so far found
b_{max1}	-	Maximum value of b at which point the second phase is to be entered if *Phase* was equal to 1
b_{max2}	-	Maximum value of b in the second phase at which point the search will be terminated
p	-	Processor Index (processor ID), $p = 0,\ 2,\ ...,\ P-1$ where P is the number of concurrently available processors
s	-	Index of Markov chain, $s = 1,\ 2,\ ...,\ S$ where S is the number of Markov chains followed (in each processors)
n	-	Iteration counter, $n = 1,\ 2,\ ...,\ N$ where N is the maximum number of iterations in each Markov chain
X_{nsp}	-	The solution at the n^{th} iteration along the s^{th} Markov chain in the p^{th} processor
α	-	Cooling schedule exponent
r	-	Index for the temperature levels in the cooling schedule
T_r	-	Temperature at the r^{th} level, $T_r = \alpha \times T_{r-1} = \alpha^r \times T_0$
Q	-	Number of iterations to be performed in each Markov chain at each temperature level
CF	-	Communication frequency. It is defined as the number of iterations to be performed by each Markov chain before communication is effected among the processors
C_{phase}	-	The number of iterations to be performed by each Markov chain before the search phase is changed form $Phase = 1$ to $Phase = 2$
rand()	-	Random number generator
Γ_i	-	The set of immediate successor items to item i
$r_{ii'}$	-	The number of item i needed by one unit of item i', where $i' \in \Gamma_i$

m_{jri} - Index of the machine type used to process operation j of part i along route r

λ_{jri} - Processing time in minutes of operation j of part i along route r

S_{ri} - Cost to set up route r of part i

$\Re_i$ - Average rework and replacement cost of one defective item of type i

H_i - Unit inventory carrying cost per period for item i

P_k - Procurement cost of type k machine,

$x_{ri}(t)$ - The production sub-lot size of item i along route r in period t

$\bar{x}_i(t)$ - The quantity of item i outsourced in period t through subcontracting

$I_i(t)$ - Inventory level of product i at the beginning of time period t

$z_{ri}(t)$ - $\begin{cases} 1, & \text{if route } r \text{ of part } i \text{ is set up for production during time period } t \\ 0, & \text{otherwise.} \end{cases}$

$\eta_{jril}(t)$ - $\begin{cases} 1, & \text{if } j^{th} \text{ operation in route } r \text{ of part } i \text{ is processed in cell } l \text{ during time period } t \\ 0, & \text{otherwise.} \end{cases}$

PS - Population size

ρ - Index for individual in a given population

f - An indicator binary variable which equal to 1 if population rejuvenation is to be accomplished; 0 otherwise.

s - Index of Markov chain, $s = 1, 2, ..., S$ where S is the number of Markov chains followed

X_{ns} - The solution at the n^{th} iteration along the s^{th} Markov chain

F	-	Interaction frequency. It is defined as the number of iterations to be performed by each Markov chain before interaction is effected among these Markov chains
M	-	Large positive number

LIST OF ACRONYMS

AI	-	Artificial Intelligence
ALC	-	Average Linkage Clustering
ANN	-	Artificial Neural Network
ART	-	Artificial Resonance Theory
AVV	-	Average Voids Value
BEA	-	Bond Energy Analysis
CAD	-	Computer Aided Design
CF	-	Cell Formation
CLC	-	Complete Linkage Clustering
CM	-	Cellular Manufacturing
CMS	-	Cellular Manufacturing Systems
CONWIP	-	Constant Work-In-Process
CRAFT	-	Computerized Relative Allocation of Facilities Technique
DCA	-	Direct Clustering Algorithm
DP	-	Dynamic Programming
DPGA	-	Distributed Parallel Genetic Algorithm
EA	-	Evolutionary Approaches
EOQ	-	Economic Order Quantity
GA	-	Genetic Algorithm
GAP	-	Generalized Assignment Problem
GP	-	Goal Programming
GT	-	Group Technology
ISNC	-	Ideal Seed Non-hierarchical Clustering
JIT	-	Just-In-Time
KB	-	Knowledge Based

LP	-	Linear Programming
LPEGA	-	Linear Programming Embedded Genetic Algorithm
LPESA	-	Linear Programming Embedded Simulated Annealing
LQP	-	Linear and Quadratic Programming
MIP	-	Mixed Integer Programming
MLCLSP	-	Multi-Item Multi-Level Capacitated Lot Sizing Problem
MPI	-	Message Passing Interface
MRP	-	Material Requirement Planning
MSPGA	-	Master-Slave Parallel Genetic Algorithm
NP	-	Non Polynomial
PGA	-	Parallel Genetic Algorithm
PMMC-SA	-	Parallel Multiple Markov Chain Simulated Annealing
PSA	-	Parallel Simulated Annealing
QPQ	-	Economic Production Quantity
ROC	-	Rank Order Clustering
SA	-	Simulated Annealing
SGA	-	Sequential Genetic Algorithm
SLC	-	Single Linkage Clustering
SMMC-SA	-	Sequential Multiple Markov Chain Simulated Annealing
SSMC-SA	-	Sequential Single Markov Chain Simulated Annealing
TS	-	Tabu Search
TSP	-	Traveling Salesman Problem
WIP	-	Work-In-Process
ZODIAC	-	Zero-One Data Ideal Seed Algorithm for Clustering

Chapter 1

Introduction

In the past few decades, there has been an increasing worldwide awareness towards productivity improvement. Advanced countries are changing to progressive concept and philosophies in the manufacturing area. The main interest of such changes lies in decentralization of the activities hitherto carried out in an autocratic and bureaucratic manner in the production units. A new style of operation and a new environment in the work place conductive to improvement in such factors as flexibility, efficiency, management-worker relation, team work and job satisfaction are becoming important for survival in the international market. Group Technology (GT) has emerged as one of the manufacturing philosophies to address such requirements. The concept is aimed at an increased productivity by exploiting the design and process similarities of the parts. Cellular manufacturing (CM) is one of the major implementations of this concept.

1.1. Cellular Manufacturing

Shorter product life-cycles, customized products, variable demand, diverse customer needs, international competitions, among others are forcing manufacturers to shift from mass production to the production of large product mixes. In order to adapt to such changes, firms are required to make their manufacturing systems efficient and flexible. Traditional manufacturing systems, such as job shops and flow lines, cannot provide the required efficiency coupled with flexibility to handle these changes.

Job shops group machines together according to the general type of manufacturing process: lathes in turning department, drill presses in drilling department, milling machines in milling department, and so forth. These type of systems are designed to achieve maximum flexibility such that a wide variety of products with small lot sizes can be manufactured. General-purpose machines are utilized because they are capable of performing many different types of operations. Figure 1.1 illustrates a job shop. When operations are completed in one department, a part has to be moved a relatively large distance to reach the next department for the next operations. In completing all the required operations, a part may have to travel the entire facility as shown in Figure 1.1. Therefore, to make the movement of parts in the system more economical, parts are transferred in batches from one department to the next. Each part in a batch must wait for the remaining parts in its batch to complete processing in a given department before it is transferred to the next department. This leads to longer production times, high levels of in-process inventory, high production costs and low production rates. In contrast to job shops, flow lines (Figure 1.2) are organized according to the sequence of operations required for a product. These systems are designed to manufacture high volumes of products

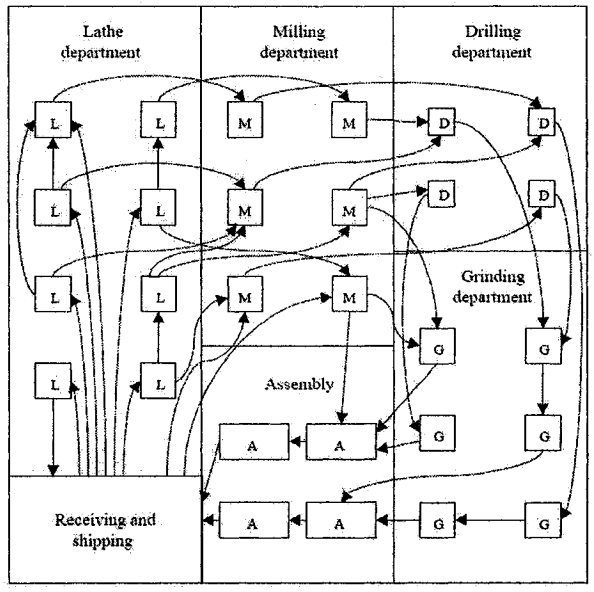


Figure 1.1: Job shop manufacturing system

with high production rates and low costs. In each line, specialized machines are dedicated to manufacture the product at high production rates. These machines are usually expensive. A large volume of the product must be produced in order to justify the investment cost of such machines. Major drawback of flow lines is the lack of flexibility. They are not suitable to produce products for which they are not designed. This is because specialized machines are setup to perform limited operations. If the design of the product is changed, a major reconfiguration of the line may be required. If new products are introduced, it may be absolutely essential to open up a new line with additional investment.

As indicated above, job shops and flow lines cannot simultaneously provide the flexibility and efficiency requirements of today's production. Cellular manufacturing, which attempts to incorporate the flexibility of job shops and the high

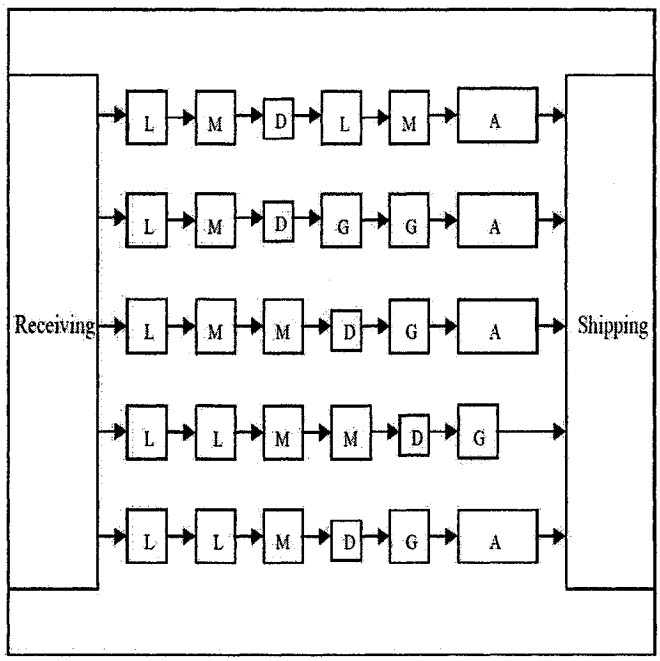


Figure 1.2: Flow line manufacturing system

production rate of flow lines, has emerged as a promising alternative to these traditional manufacturing systems. CM is a production approach aimed at increased production efficiency and system flexibility by utilizing processing similarities of the parts. It involves grouping similar parts into part families and the corresponding machines into machine cells. This results in the organization of production systems into more or less self-contained and self-regulated groups of machines such that each group of machines undertake an efficient production of a family of parts. Figure 1.3 illustrated a particular CM layout. The design of CM involves the formation of part families based upon their similar processing requirements and the grouping of machines into manufacturing cells to produce the part families. A part family is a collection of parts which are similar either because of geometric shape and size or similar processing steps required in their manufacture. A manufacturing cell consists

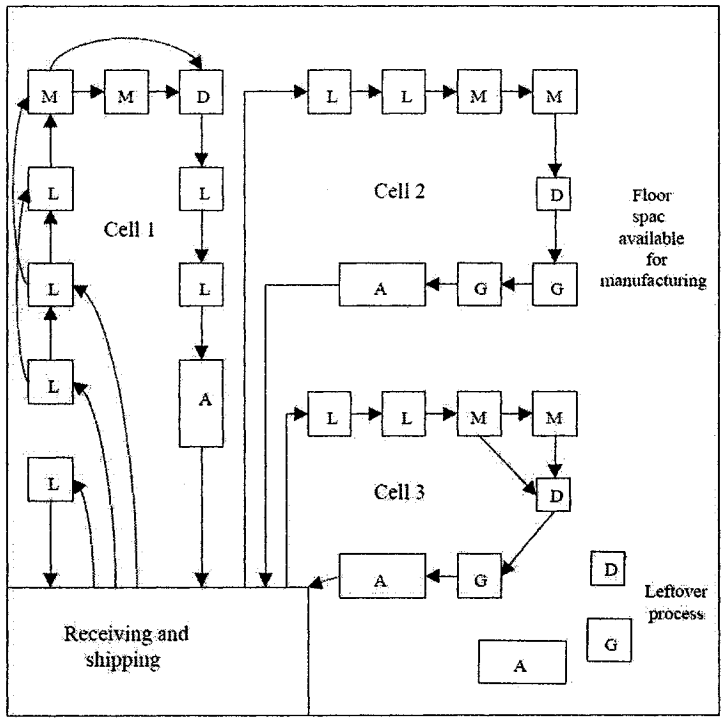


Figure 1.3: Cellular manufacturing system

of several functionally dissimilar machines which are placed in close proximity to one another and dedicated to the manufacture of a part family. The manufacturing cells should be as independent as possible in order to satisfy the required performance criteria, for example the number of inter-cell movements.

1.2. Advantages of Cellular Manufacturing

The wide acceptance and the successful implementation of CM have been discussed and surveyed in [159] and [112]. According to these two papers and other related reports, the advantages derived from successful implementation of CM can be summarized as follows.

1. **Reduction in tooling and setup time:** A manufacturing cell is designed to handle part families having similar manufacturing requirements. For this reason, it is logical to design a group of jig/fixture to accommodate the tooling requirement of the part family. Usually these group jig/fixtures are designed to accept all members of the family accompanied with adapters that meet the specific needs and differences of the parts in the family. With a group jig/fixture, there is no need to design each individual tooling for each part. Most adapters are very inexpensive compared to the regular jig/fixtures. Therefore, obviously this save a great deal of tooling costs. In addition to this, the use of generic fixtures for the part family can reduce the time required for changing fixtures and tools.

2. **Simplified production and inventory control:** Several benefits accrue to a company's production and inventory control function as a consequence of CM. Production scheduling is simplified with CM. In effect, grouping of machines into cells reduces the number of production centers that must be scheduled. Grouping of parts into families reduces the complexity and size of the parts scheduling problem. Because of more efficient material handling within machine cells, manufacturing lead time and work-in-process (WIP) are reduced making the inventory control much simpler. Reduced setup cost makes smaller lot sizes economical. Smaller lot sizes are directly related to the reduction of WIP.

3. **Reduced material handling cost and flow time:** In CM, each part is processed completely within a single cell (where possible). Thus, part travel time and distance moved between consecutive operations is minimal. Thus, a reduction in material handling cost and flow time is obtained.

4. **Reduced throughput times:** In a job shop, parts are transferred between machines in batches. However, in CM each part is transferred immediately to the next machine after it has been processed. Thus, the waiting time is reduced substantially.

5. **Improved product quality:** Since parts travel from one station to another as single units, they are completely processed in a small area. The feedback is immediate and the process can be stopped when things go wrong.

6. **Better employee satisfaction:** The machine cell often allows parts to be processed from raw material to finished state by a small group of workers. The workers are able to visualize their contribution to the firm clearly. This tends to cultivate an improved worker attitude and a higher level of job satisfaction. Employee/worker benefits also include worker flexibility, importance of social group, reduced frustration and improved status and job security.

1.3. Research Objectives

The design of CM can be broadly divided into two major phases [19]. The first phase of CM design is cell formation while the second phase consists of the system design of each of the individual cells. The first phase is the major concern of this thesis work. The effectiveness of this phase depends heavily on the consideration of several factors. From our review of recently published cell formation methods we noticed that these factors include:

1. Alternative routing
2. Demand fluctuation
3. Dynamic cell reconfiguration
4. Workload balancing
5. Lot-splitting
6. Types of tools required by a part

7. Types of tools available on a machine
8. Machine proximity constraint
9. Sequence of operation
10. Setup cost/time
11. Cell / part family size constraint
12. Movement of parts (material handling cost)
13. Machine capacity
14. Presence of identical machines
15. Machine investment cost
16. Subcontracting cost
17. Tool consumption cost
18. Unit operation time
19. Operation cost
20. Product structure (bill of materials)
21. Production planning

Our review also showed that individual methods published recently addressed only a limited subsets of these factors. The objective of this research is, therefore, to develop a comprehensive mathematical model that will cover the majority/all of these factors. The solution of such a model is NP-hard. The general branch and bound approach used by most off-the-shelf optimization software cannot be used to solve real size problems of this nature. In order to solve real size problems efficiently, a heuristic search method based on a selected meta-heuristic as genetic algorithm, simulated annealing or tabu search will be developed. In this endeavor, the possibility of developing an interactive solver between simplex algorithms and search method and the use of parallel computing opportunities will be also explored.

1.4. Research Approach

To achieve the development of the comprehensive CM design methodology, the research approach consists of the following steps:

1. Identify the factors that should be considered during the design of CM by

making a review of published articles.

2. Formulate a comprehensive mathematical model for dynamic production requirements incorporating the factors identified in step 1 above.
3. Convert the mathematical formulation into a format that can be recognized by a selected off-the-shelf optimization package.
4. Generate problem instances to be used for validating the developed model. (Data will be generated in the range of the data found in published articles and case studies)
5. Solve the problem instances optimally using the selected software package and analyze results.
6. Evaluate the potential benefits of the model gained through the consideration of the various factors incorporated in it.
7. Develop a heuristic approach to solve the model efficiently for large data set.
8. Write a computer code of the developed heuristic using a selected programming language.
9. Validate the computational efficiency of the developed heuristic by comparing its performance to that of the selected software package.
10. Draw conclusions and discuss the directions for future works.

1.5. List of Publications

In this section is the list of papers and conference presentations that I co-authored during my stay in Concordia University as a PhD student.

Accepted Journal Paper

1 Defrsha, F. M., and Chen, M., (2005). Machine Cell Formation Using a Mathematical Model and a Genetic Algorithm Based Heuristic. Accepted in July 2005 for publication in the *International Journal of Production Research.* ***Reference No. IJPR 40372***

2 Defrsha, F. M., and Chen, M., (2005). A Comprehensive Mathematical Model in the Design of Cellular Manufacturing System. Accepted in October 2005 for publication in the *International Journal of Production Economics.* ***Reference No. IJPE 425 ACMA***

3 Hu, B., Chen, M., and Defersha, F. M., (2005). An Integrated Method for Multi-Objective Cell Formation in Cellular Manufacturing Systems. Accepted in November 2005 for publication in the *International Journal of Manufacturing Technology and Management.* ***Reference No. IJMTM-MCE-012***

Refereed Conference Proceeding

4 Defersha, F.M., and Chen, M., (2004), Designing Cellular Manufacturing Systems: A Genetic Algorithm Approach in *Advances in Dynamics, Instrumentation and Control*, Editors: C-Y. Su, S. Rakheja, E. Wang and R. Bhat (***World Scientific Publishing Co.***) pp.387-396

5 Cao, D., Chen, M., and Defersha, F.M., (2005), Lot streaming and operation regrouping considering alternative process routes and production quality in the proceedings of the International Conference on Production Research (ICPR) Salerno, Italy, July 31 - August 4, 2005

Conference Presentation

6 Defersha, F.M., and Chen, M., (2005), A Coarse-Grained Parallel Genetic Algorithm for Cellular Manufacturing System Design. Optimization Days 2005, Montreal, CANADA, May 9 - 11, 2005

7 Defersha, F.M., and Chen, M., (2005), A Parallel Simulated Annealing Algorithm for Cellular Manufacturing System Design. 47th Annual Conference of the Canadian Operational Research Society, Halifax, CANADA, May 16 - 18, 2005

8 Hu, B., Chen, M., and Defersha, F.M., (2005), An Integrated Method for Cellular Manufacturing Systems Design and Operation. 17th Triennial Conference of the International Federation of Operational Research Societies, Honolulu, Hawaii, United States, July 11 - 15, 2005

Journal Paper Under Review

9 Cao, D., Defersha, F. M., and Chen, M., — Lot Streaming and Product Quality Improvement in Cellular Manufacturing Systems. Submitted to the *International Journal of Production Research* on February 10, 2005.

10 Defrsha, F. M., and Chen, M., — A Parallel Multiple Markov Chain Simulated Annealing for Cellular Manufacturing System Design. Submitted to *Computers in Industrial Engineering* on October 17, 2005.

Refereed Conference Proceeding Under Review

11 Koganti, R., Chen, M., and Defersha, F.M., (2006), Design for Integrated Assembly and Disassembly of Automotive Products. Under review for the proceedings of SAE World Congress, Detroit, Michigan, USA, April 3-6, 2006

Accepted Conference Abstracts

12 Defersha, F.M., and Chen, M., (2006), An Integrated Approach in the Design of Cellular Manufacturing and Production Planning for Dynamic Production Requirements. To be presented in the upcoming Optimization Days/CORS 2006 joint conference.

13 Defersha, F.M., and Chen, M., (2006), A Multi-level Multi-item Capacitated Lot Sizing by considering the Impact of Run Length on Product Quality. To be presented in the upcoming Optimization Days/CORS 2006 joint conference.

14

1.6. Outline of Document

The remainder of this thesis is organized as follows. In chapter 2, we present a review of the literature in the design of cellular manufacturing systems. Chapter 3 provides a comprehensive mathematical model developed for designing a dynamic cellular manufacturing system. Numerical examples are provided by solving the model using off-the-shelf optimization software to illustrate the features of the model and its potential benefits. In Chapter 4 and 5, search heuristics based on genetic algorithm and simulated annealing respectively, are presented to solve the proposed model. The heuristics were implemented both on single processor machine and in a parallel computing environment. Chapter 6 presents a second mathematical model addressing an integrated cell formation and production planning problem in MRP environment. The impact of lot size on product quality is also incorporated in the developed model. In Chapter 7 we developed a hybrid search heuristics that integrates the simplex method in linear programming with genetic algorithm in order to solve integrated cell formation production planning model efficiently. The

basic genetic algorithm uses the simplex method to solve a linear programming subproblem corresponding to each individual in the population in each generation. In this chapter, a similar hybrid heuristic is also presented using simulated annealing. Summary, conclusion and future research are presented in Chapter 8.

Chapter 2

Literature Review

2.1. Introduction

Since it was asserted for the first time, the cell formation problem has grown into an area in which much research has been conducted. Various methods of cell formation have also been proposed. Comprehensive summaries and taxonomies of studies devoted to part-machine grouping problems are presented by many researchers (e.g., [158, 80, 148, 66, 131]). This chapter presents a review of related literatures in the area of machine cell formation in cellular manufacturing system. Recently published articles are considered in order to identify the manufacturing attributes that should be considered during the design of CM.

2.2. Taxonomy of CM Design Methods

At the highest level, methods for part family/machine cell formation can be classified as design-oriented or production-oriented. Design-oriented approaches group parts into families based on similar design features while production-oriented

techniques aggregate parts requiring similar processing. Classification and coding schemes are design-oriented tools that can be used to implement GT applications [59, 83]. An overview of classification and coding is presented by Askin and Vakharia [7]. Analysis of codes facilitates the rationalization of the design process, rapid prototyping, the development of new parts, and to a certain extent can be used for machine cell formation. Since part codes are assigned based upon physical geometry, parts having similar design features have similar codes providing a week connection between part features and machine grouping [74, 158]. This makes the application of classification and coding to machine cell formation very limited. This can be seen by the fact that the large number of CM designed methods proposed during the late decades are not based on classification and coding. They are production-oriented approaches. The production-oriented approaches can be further classified into: Cluster Analysis, graph partitioning, mathematical programming, AI-based approaches, and heuristics [47, 69, 66].

2.2.1. Cluster Analysis Procedures

Cluster analysis is composed of many diverse techniques for recognizing structure in a complex data set. The main objective of this statistical tool is to group either objects or entities or their attributes into clusters such that individual elements within a cluster have a high degree of "natural association" among themselves and that there is very little "natural association" between clusters. Clustering procedures can be classified as: 1) array-based clustering techniques, 2) hierarchical clustering techniques, and 3) non-hierarchical clustering techniques.

Array-based clustering

Array-based clustering techniques are the most cited methods in CM research. In these methods, the processing requirements of parts on machines are represented by an incidence matrix, referred to as the machine-part matrix. The machine-part matrix has zero and one entries (a_{ki}). A 1 entry in row k and column i ($a_{ki} = 1$) of the matrix indicates that part i has one or more operations on machine k, whereas a 0 entry indicates that it does not. These techniques try to allocate machines to groups and parts to families by appropriately rearranging the order of rows and columns to find a block diagonal form of the $a_{ki} = 1$ entries in the machine-part matrix. Bond Energy Analysis (BEA) [99], Rank Order Clustering (ROC) [76] and Direct Clustering Algorithm (DCA) [24] are typical examples of array-based clustering methods. Chu and Tsai [37] conducted a comparative study of BEA, ROC and DCA. They concluded that the BEA significantly outperformed the other two regardless the nature of the initial part/machine incidence matrix.

Matrix manipulation methods are very simple and computationally efficient to apply to the part/machine incidence matrix [38]. Their popularity stems from this fact. However, these algorithms generally do not take into account other types of manufacturing data such as production volume, machine capacity, cost of machines and cell size limits. These techniques also usually required visual inspection of the output to determine the composition of the manufacturing cell. Such visual inspections are very difficult for matrixes of practical sizes. In addition to these limitations, the array based methods have the disadvantages of being dependent on the initial configuration of the zero-one matrix [139]. They are not able to provide disjoint part-families and machine-cells for ill structured matrix.

Hierarchical clustering

Hierarchical clustering techniques operate on an input data set described in terms of a similarity or distance coefficient between each pair of individuals. This can be represented as a lower triangular matrix since the similarity between individuals is commutative. Using the data contained in this matrix the technique produce a hierarchy of cluster or partition in a progressive manner. Thus, the data are not partitioned into a particular cluster in a single step. This clustering technique is subdivided into divisive and agglomerative methods. Divisive methods run from a single cluster containing all objects to n clusters each containing a single object. On the other hand, agglomerative methods run from n clusters each containing a single object to a single cluster containing all objects. Hierarchical clustering may be represented by a two dimensional diagram known as dendrogram which illustrates the fusions or divisions made at each successive stage of analysis. An example of such a dendrogram is given in Figure 2.1.The cell designer must choose a similarity level or threshold in order to define the number of clusters. As the threshold increases, the number of cells in creases while the size of each cell decreases.

Stanfel [142] seems to be the only researcher who applied the divisive approach to cell formation. Agglomerative techniques are more commonly used. These techniques generate a series of partitions of the data, P_n, P_{n-1}, ..., P_1. The first P_n consists of n single object 'clusters', the last P_1, consists of single group containing all n cases. At each particular stage the method joins together the two clusters which are closest together (most similar). Differences between methods arise because of the different ways of defining distance (or similarity) between clusters. This results in a different agglomerative techniques as single linkage clustering (SLC), average linkage clustering (ALC) and complete linkage clustering(CLC). The defining feature of

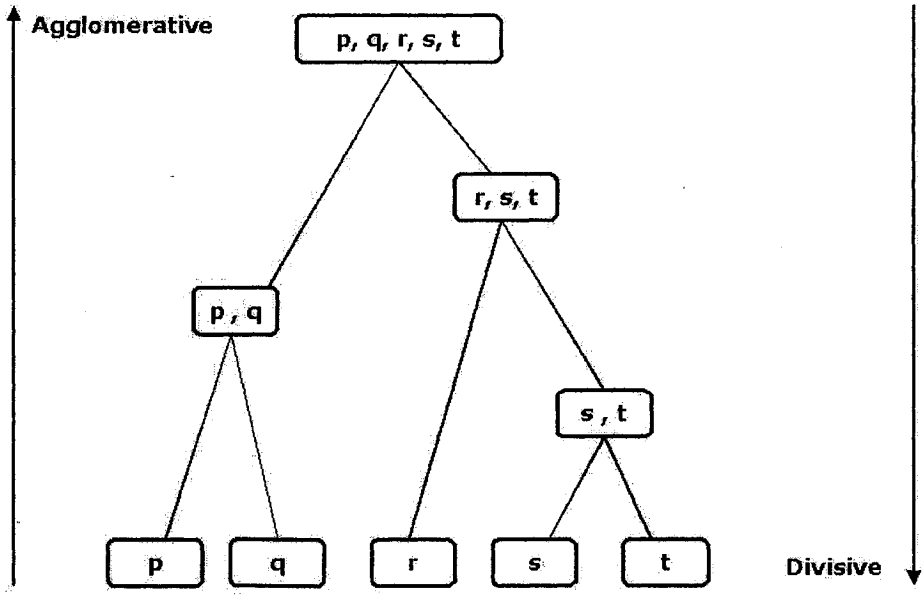


Figure 2.1: An example of dendrogram

SLC is that distance between clusters is defined as the distance between the closest pair of objects where a pair of objects is constructed by taking one element from each of the two clusters. McAuley [98] was probably the first author who applied hierarchical clustering to solve cell formation problem using SLC. This technique has a sever chaining problem, which means that two cluster can be grouped based merely upon a single bond between one machine in ache cluster [98, 105, 51]. The chaining problem can lead to improper machine assignments in the groups. To help reduce the chaining problem, Seifoddini [129] applied ALC. In this technique, the similarity between two clusters is defined as the average of the similarities of each pair of objects taken from each cluster. CLC further reduces the chaining problem by selection the minimum similarity coefficient as the in-between cluster relationship instead of the maximum or average [105, 51].

As it has been stated at the outset, the input for hierarchical clustering techniques is the similarity coefficients between each pair of individuals in the data set. Thus, a choice of similarity measure is an important aspect in the use of hierarchical clustering technique for CM design. Similarity coefficients can incorporate manufacturing data other than just the binary part/machine incidence matrix. A number of research papers have used different types of similarity and dissimilarity coefficients for determining part families. McAuley [98] was the first researcher to apply the Jaccard similarity coefficient to the cell formation problem. McAuley calculated the Jaccard similarity coefficient for each machine type pair. Selvam and Balasubramanian [132] developed a dissimilarity measure based on operation sequences. Dutta *et al.* [42] developed a dissimilarity coefficient to cluster parts. Choobineh [35] proposed a similarity measure which uses the manufacturing operations and their sequences in the first stage. Gunasingh and Lashkari [48] suggested a similarity index which expressed the capability between two machines in processing a set of parts that need both machines. The capability of a machine is defined in terms of the tools available to it and tooling requirements of the parts. Tam [146] also proposed a new similarity coefficient based on the similarity of operation sequences. Gupta and Seifoddini [51] suggested a new similarity coefficient which was based on the idea that necessary production data should be incorporated in the early stages of the machine-component grouping process. They considered processing requirements of parts, pair wise average production volume and unit operation time as new production parameters. Vakharia and Wemmerlöve [149] proposed a new coefficient for use in the clustering process by considering the within-cell machine sequence and machine loads. In this similarity coefficient, the proportion of machine types required by two parts in the same order is measured. Kusiak and Cho [82] proposed two new similarity measures in which one is a binary measure that indicates

whether one part's process plan is a subset of another part's process plan. The other similarity measure is a modified version of the first which is to be used when the value of the first similarity measure would have been zero. Gupta [50] suggested a new similarity coefficient which required that alternative routing of parts should be considered while calculating the pair wise similarity coefficient between machines. Kamrani and Parsaie [68] proposed a weighted dissimilarity coefficient based on a disagreement measure of both design and manufacturing attributes between two parts. Jeon *et al.* [65] developed a similarity coefficient which considers alternative routes during machine failure. Yasuda and Yin [167] proposed a new dissimilarity measure for cell formation problem based upon the calculation of an average voids value (AVV). The AVV indicates the average number of newly produced voids when a pair of machine groups are combined.

Non-hierarchical Clustering

Non-hierarchical clustering methods are iterative methods and they begin with either an initial partition of the data set or the choice of a few seed points. In either case, one has to decide the number of clusters in advance. After initial clusters are formed reallocation occurs iteratively according to some optimality criterion. In contrast to the hierarchical method, this technique permits objects to change group membership through the cluster formation process. As an example, let us consider one of the most popular non-hierarchical clustering technique, namely the k-mean algorithm. Suppose we have n feature vectors X_1, X_2, ..., X_n all belonging to the same class C. We assume that they belong to k clusters such that $k < n$ and initialize the means (seed vectors) μ_1, ..., μ_k as the centers of the k clusters to be formed. One of the ways to do this is just to assign random vectors to them. We then determine the membership of each X by taking the $||X - \mu_i||$. The minimum

distance determines X's membership in a respective cluster. Hence, one vector will only belong to one and only one cluster. Once the membership of each X is determined, new values of means $\mu_1, \ldots, \mu_k$ will be determined as the centroid of the generated k cluster. With these newly calculated mean vectors, the membership of each X will be recalculated. Some vectors will then be relocated. This process will continue until no changes in the membership occur. As it can be seen from this process, in non-hierarchical clustering techniques, there is no need to process a tree-like structure to determine the final clusters. Clusters are formed by relocating objects to predetermined number of clusters. Thus, only k partition are processed during the entire process. Notice that, the agglomerative hierarchical clustering starts with n partitions each of them containing only one object. Then, by successive merging it ends up with one partition containing all the objects. This successive merging is represented by an inverted tree like structure - a hierarchy of clusters. The final cluster are then identified by examining the tree with certain threshold value of similarities. The advantage of non-hierarchical clustering over hierarchical clustering is that there is no need to examine a tree like structure to determine the final cluster which is a difficult job for large data set; a similarity or distance matrix does not need to be computed or sorted [4]; more natural cluster tend to be formed because data members are not permanently bound to a group during the cluster identification process [25]. The obvious disadvantage is that the number of clusters must be specified a prior, potentially forcing some natural clusters to merged or partitioned. This may require reprocessing the data for different cluster numbers to evaluate the sensitivity of the result.

Lemoine and Mutel [88] proposed a non-hierarchical clustering technique for automatic recognition of production cells and part families in the design of CM. Chandrasekharan and Rajagopalan [25] developed an algorithm called Ideal Seed

Non-hierarchical Clustering (ISNC) for either machine cell or part family formation. This technique uses an evaluation criterion called group efficiency, which measures intercell movement and within-cell machine utilization. To overcome the limitation of specifying the number of clusters a priori, the problem is first formed as a bipartite graph. Then, a theoretical upper limit on the maximum number of independent part families or machine cells is developed. Another non-hierarchical clustering technique called ZODIAC (zero-one data ideal seed algorithm for clustering) was developed by Chandrasekharan and Rajagopalan [26] which is a much improved and expanded version of ISNC [25]. ZODIAC was designed to simultaneously identifies the part families and machine cells. Srinivasan and Narendran [140] showed that the initial seed selection of ZODIAC can still lead to a collapse of some beneficial clusters or numerous groups with singleton members. Also the minimum rectilinear distance used as the basis for clustering does not truly represent the machine processing that is required by individual parts. To overcome these limitations of ZODIAC, Srinivasan and Narendran [140] developed an algorithm called GRAFICS which generate initial seeds from an assignment problem, which maximizes the similarity between machines. Each of the sub-tours is identified and used as initial seeds in a non-hierarchical clustering algorithm using the maximum density rule as the clustering criterion. Recently, Nair and Narendran [108] presented non-hierarchical clustering algorithm which clusters machines and parts on the basis of sequence data. Ohta and Nakamura [111] extended this work and proposed a new non-hierarchical clustering algorithm for cell formation with reduction in setup times between machines in the same cell.

2.2.2. Graph Partitioning Approaches

A graph $G(V, E)$ is a structure consisting of a set of vertices $V = \{v_1, v_2, \ldots v_m\}$ and a set of edges $E = \{e_1, e_2, e_3 \ldots\}$. All vertices are represented by circles (nodes) and edges as lines in a graph. Cell formation graph partitioning methods treat the machines and/or parts as vertices and the processing of parts as arcs connecting these nodes. These models aim at obtaining disconnected subgraphs from a machine-machine or machine-part graph to identify manufacturing cells. Rajagopalan and Batra [123] were among the first to apply a purely graph theoretic approach to the cell formation problem. Machines are represented by vertices. The edges between each pair of vertices are assigned the Jaccard's similarity coefficients to represent the strength relationship of a machine pair. The authors assumed prespecified threshold value of the similarity coefficient to avoid using a weak relationship; i.e., no edges exist between a pair of machines whose Jaccard's similarity is below a threshold value. After all allowable edges have been introduced, cliques are formed. These cliques are then merged to create cells so that intercell moves are minimized. An upper limit on cell size determines the maximum number of machines in each partition. During the process high and balanced machine utilization are strived for and machine loads are used to determine the number of machines of a given type needed for each cell. Witte [161] use this approach with different similarity coefficients. King and Nakornchai [77] suggested that cell formation could be represented as a bipartite graph by letting the parts and machines represent the two sets. An edge between the sets represents the processing of a part on a machine. Faber and Carter [43] developed a graph theoretic algorithm for grouping machines and parts into manufacturing cells by converting the machine similarity matrix into a cluster network. The cluster network is partitioned into cells by solving a minimum cost

flow problem. Vohra *et al.* [153] proposed a network approach using a modified Gomory-Hu algorithm to find the minimum intercellular interaction. Askin and Chiu [6] proposed a cost-based mathematical formulation and heuristic graph partitioning solution procedure for the cell formation problem. Hertz *et al.* [54] applied graph theory approach to three types of economic decisions in CM: subcontracting, machine duplication and intercell moves. The problem is formulated as a minimum weighted node covering problem in a hypergamy, and they showed that it can be solved in polynomial time by finding a maximum weighted stable set in a bipartite graph. Recently, Mukhopadhyay *et al.* [106] applied Hamiltonian chain graph theoretic approach to group technology. Hamiltonian path was modified by using dummy edges for better accessibility in order to arrive at a block diagonal solution to a given problem. They used the number of parts requiring a given machine pair as a measure of strength.

2.2.3. AI-based Approaches

Researchers have increasingly applied artificial intelligence (AI) techniques to the cellular manufacturing problem. These techniques include syntactic pattern recognition, expert systems, fuzzy mathematics, neural network, evolutionary approaches (EA), simulated annealing (SA) and tabu search (TS).

Syntactic pattern recognition

Syntactic pattern recognition has been commonly used in the field of AI for parsing strings in natural language processing. Wu *et al.* [164] applied this technique to design of manufacturing cells given the machine sequence data of the parts. Machine sequences are treated as strings which are used to form part families. The

method derives a dendrogram from which multiple clusters with different number of cells can be derived. It is possible to generate alternative groupings of some parts from the dendrogram allowing the decision maker to balance load among cells to a certain extent. Classification of a new/modified part into an existing machine cell can be done by matching the parts machine sequence with that of a cells pattern.

Expert systems

Kusiak [81] proposed a method based on expert systems, the most successful and developed branch of AI. Kusiak starts with the machine- part incidence matrix modified to include the processing time of part j on machine i instead of the binary (0, 1) data. The procedure to form part families and machine cells uses a knowledge-base and a clustering algorithm closely interacting with each other. Each iteration determines machine cells and selects part families based on a heuristic which takes into account variables such as production costs, material carrier, etc. The assignment is checked for satisfaction of three or four meta-constraints such as availability of processing time in a cell and maximum number of machine cells. If any of the constraints is violated, appropriate rules from the knowledge-base are fired to correct the situation. For example, if the processing time constraint of a cell is violated, new machines are introduced or alternative processing plans are considered. Recently, Luong [96] developed an knowledge-based (KB) system that attempts to make recommendations of system feasibility, cell formation techniques and cell types during the conceptual design of CM. The recommendation for system feasibility is based on the production quantity and product variety ration ($Q\backslash P$). The cell formation techniques that the KB system will recommend are either classification and coding or production flow analysis. The types of cells that are recognized by the KB system for recommendation are classified as product focused, process-focused, general-purpose

and hybrid cells.

Fuzzy Logic

Most clustering methods assume that part families are mutually exclusive and collectively exhaustive [168]. While some parts definitely belong to certain part families, it is not always clear which families is appropriate. Xu and Wang [166] applied a fuzzy mathematics to this problem. Part features (e.g., length, diameter) are transformed by membership functions into fuzzy numbers. The membership function of each feature is designed such that the resulting fuzzy number is able to differentiate parts according to the feature's processing needs. The fuzzy numbers are then used to construct a similarity coefficient matrix. A threshold value is used to specify the minimum value of the similarity coefficient for a part to be in the same family. Narayanaswamy *et al.* [109] developed a non-binary machine component matrix wherein each entry indicates the suitability of a machine to process a part in dealing with alternative routings. Fuzzy logic concepts were used to consider the interaction between part features and the importance of certain part features. The uncertainties in the dimensional tolerance and processing times were also considered for the determination of the overall suitability of a machine to process a part. They concluded that existing fuzzy rank order and fuzzy single linkage clustering algorithms can be applied to this non-binary machine-component matrix for subsequent part family and machine group formation. Güngör and Arikan [49] applied fuzzy set theory and developed an algorithm which considers both design and manufacturing attributes and operation sequences as input parameters to formulate the cell formation problem. These parameters are fuzzified using membership function concept. Using the fuzzified input parameters, IF-THEN decision rules are applied to determine parts relationship as fuzzy sets. Crisp values of the parts relationship is

then calculated by the defuzzification step. After these steps, the defuzzified parts relationship chart is used as an input to the traditional cell formation procedure namely Single Linkage Clustering (SLC). Spatial arrangement of machines in each cell are found by using the Computerized Relative Allocation of Facilities Technique (CRAFT). Recent repots on the use of fuzzy mathematics for CM design can be found in [67] and [95].

Numeral Network

Artificial neural networks have been applied successfully to many manufacturing areas [169]. These techniques have substantial potential for application to GT because of their ability to learn through a training process by recognizing patterns and memorizing special features associated with input data. Several researchers have applied artificial neural networks to the classification and coding and related problems in GT. Awwal and Karim [9] applied a hopfield neural network to recognize the shapes of the parts in the form of binary images. Four part shapes were used to train a neural network and nine partial inputs shapes were provided to test the recognition capabilities of the network. Kaparthi and Suresh [70, 71] have applied the ANNs for classification and coding of rotational parts using three-digit part description, whereas Liao and Lee [90] presented an automated GT coding and part-gased CAD system. The above mentioned ANN methods are for design-oriented GT application. ANNs have been also applied to a production-oriented methods to determine machine cells and part families. Malave [97] applied a modified version of the Hebbian learning rule to the cell formation problem, while others have applied other unsupervised neural learning algorithms such as competitive learning [36, 104, 150] and Kohonen nets [150]. Several researchers used the neural network classifier based on an unsupervised learning model by Carpenter and Grossberg [21] and its variants in

[32, 84, 39]. Another variant of the ART model, Fuzzy-ART, handles both analogue and binary valued inputs while utilizing new learning laws [18, 145].

Evolutionary and Local Search Based Algorithms (EA, SA & TS)

Evolutionary approaches (EA), simulated annealing (SA), and tabu search (TS) are very efficient search algorithms gaining popularity in solving a wide variety of engineering problems. In literature, these approaches are classified under artificial intelligence techniques.

EAs has shown an interesting potential in the engineering fields. The optimization strategies used, based on a population of solutions, have led an increasing number of researchers to publish articles that address several types of problems encountered in the area of manufacturing systems [114]. EAs have been applied to solve part-machine problems by several researchers separately. Pierreval and Plaquin [116] address the formation of cells to minimize the traffic of parts. In their approach, solutions are coded through integer strings where the value of an element represents the cell number containing a machine. The crossover operator is aimed at passing the similarities between the two parents on the offspring and builds only feasible solutions. Caux and Pierreval [23] propose a method to form cells composed both of machines and operators, taking into account the operators skills. They avoid assigning an operator to a cell containing machines on which this operator is not skilled. Training costs can be considered to increase the quantity of work performed in each cell. The objective is then to minimize the training costs regarding to the quantity of work. In this case, the EA manages a population of solutions where each solution indicates the grouping of machines and operators into cells. King *et al.* [78] propose a GA for the cell formation problem that takes into account alternative routings. The vector coding solutions contains two kinds of information: the first

part models the assignment of machines to cells and the second part indicates the chosen routing for each part. Kazerooni *et al.* [73] solved a part-machine problem using GA taking into account production volumes, alternative routings and process sequences. Cheng *et al.* [33] addressed the formulation of the part-machine problem through a traveling salesman problem (TSP) and developed a genetic algorithm solution procedure. A flow method then optimally assigns, within each cell, operators to machines. Multi-criteria cell formation is addressed using an EA in [115]. In this case, the selection process of the EA is based on a tournament and uses the notion of Pareto-optimality (or non-dominated individuals). In their article, the multi-criteria approach is illustrated on an example where the inter-cell traffic is minimized and the cell workload is balanced. The evaluation function is the number of intercellular movements. The inter-cell minimization problem also including specific constraints. The cell design has to be done, such as, for example, some machines must be in the same cell or conversely in two different cells. Rao *et al.* [125] addressed the machine-grouping problem using a EA and the layout problem using a placement algorithm. The EA provides a population of solutions for the machine grouping problem, then for each solution, they apply a greedy algorithm in order to obtain a layout, which meets the physical constraints (e.g. a machine cannot be moved). The total distance traveled is then computed for each solution and, obviously, the objective is to minimize this distance. A genetic algorithm approach based on a coding specification scheme, as used in Group Technology, is proposed by [87] to identify part families. The GA aims at creating part families by maximizing the similarities among parts of a same family. The similarity between two parts depends both of design attributes (shape, tolerances, etc.) and manufacturing attributes (operation sequence, process, etc.). In certain cases, it may not be relevant to structure the total workshop in a cellular organization. In a hybrid organization, in addition to group technology cells

certain machines are grouped into functional cells, composed only of machines of the same type. Genetic algorithm approaches for designing cellular manufacturing systems with dynamic part populations can be found in [160] and [107]. In these articles, the cell formation problem is considered over a multiple planning periods. System reconfiguration is taken into consideration in order to cope with the changes in demand. Mungwattana [107] considered both deterministic and probabilistic demand fluctuation. Viguier and Pierreval [151] propose a multi-criteria evolutionary programming approach to define a hybrid organization, taking into account several possible routings for the parts. The solutions representations are vectors that contain both information related to which cell contains which machine, and which routing is assigned to which part. Mutations can change the assignment of machines or the routing of parts. Zolfaghari and Liang [171] developed a GA for solving a machine/part grouping problem by considering processing times, lot sizes and machine capacities. In their article, they provided a generalized group efficiency index (Γ_g) used as a fitness function to guide the genetic search. Γ_g quantifies the density of operations within cells (within cell utilization) and the intercell moves in a grouping problem by utilizing the processing time and lot sizes.

SA is another successful stochastic approaches in the class of AI based methods. Kirkpatrick *et al.* [79] initially presented the simulated annealing algorithm, which attempts to solve hard combinatorial optimization problems through controlled randomization. Since then the algorithm has been applied to many optimization problems in a wide variety of areas, including cell formation problems [27, 1, 135, 143, 107, 157, 165]. Mungwattana [107] developed a problem specific heuristic that generated an initial configuration. Simulated annealing is then employed to improve the initial cell configuration generated by the heuristic. His work considered system reconfiguration to cope with changes in demand over the planning horizon. Wang

et al. [157] applied simulated annealing to solve inter- and intra-cellular facility layout of CM simultaneously assuming the demand rate varies over the product life cycle. Cell formation techniques using TS can be found in [93], [92], [94], [41, 20], and [31]

2.2.4. Mathematical Programming

Mathematical programming approaches are widely employed in the design of CMSs. These techniques can be classified as linear programming (LP), linear and quadratic programming (LQP), dynamic programming (DP) and goal programming (GP) [131]. They offer distinct advantage over other cell formation techniques as they can easily incorporate a number of design logics in their objective and constraint functions. Mathematical formulations enable cell designers to easily incorporate ordered sequences of operations, alternative process plans, multiple and non-consecutive part operation on the same machines, operation cost, machine holding cost, setup and processing times, the use of multiple identical machines, workload balancing, production planning issues as well as outsourcing of parts. These formulations also suffer from critical limitation of being computationally intractable for realistically sized problems. Large scala problems typically require heuristics and approximate methods such as Lagrangian relaxation with subgradient optimization, simulated annealing and genetic algorithms. Purcheck [1974, 1975] was among the first to apply linear programming techniques to the GT problem. As an optimization technique, the objective in cluster analysis is to maximize the total sum of similarities between each pair of individuals (machines or parts) or to minimize the distances between each pair.

The classical clustering p-median model is used to cluster n parts (machines)

into p part families (machine cells). Constraints specify that each part can belong to only one part family and the required number of part families is p. A part can be only assigned to a part family that has been formed. Solutions obtained are optimal for a specified p, requiring that all values of p be evaluated to find the minimum objective function [141]. The p-median model assumes that each part has only one set of machining operation, i.e., there are no alternative process plans. The generalized p-median model [80] relaxed this assumption and considerers the presence of alternative routings. Since Kusiak suggested the generalized p-median model as a solution methodology for solving cell formation problem, many authors have reported successful applications to cell configuration with slight modifications over Kusiaks p-median model [126, 85, 152, 156, 40]. However, real applications of the p-median model were severely restricted due to the two major factors: problem size and software type [163]. First, regarding the problem size, let us suppose an $m \times n$ machine-part incidence matrix. Then m^2 binary variables are needed to implement the previous p-median models of machine cell formation and the CF problem is NP-complete [77]. Therefore, it takes prohibitive computation time to even solve medium-sized CF problem by using the previous p-median models without proper acceleration schemes. The performances on large scale CF problems with 40 or more machines has been little reported in literature. Secondly, with respect to the software type, the HYPER LINDO program is the most popular software for solving mathematical formulations of the CF problem [85]. However, it has a limited capability of solving small-size CF problems with less machines. To resolve the drawbacks of the original version by Kusiak [80], Kaparthi and Suresh [72] augmented the original p-median model so that it can accommodate larger CF problems with the merit of much fewer binary variables by adopting the similarity coefficient

defined between parts. But it still requires too many constraints even on medium-sized CF problems. Wang and Roze [156] and Islam and Sarker [61] proposed more efficient versions with fewer constraints by introducing an upper limit on the cell size which indicates the maximum number of machines allowable to each cell. But those modified versions of the p-median model did not suggest any efficient schemes for speedy implementation of the proposed linear integer programming formulations [163]. Won [162] used the binary variable reduction technique to reformulate the classical p-median model so that the resulting one contains fewer binary variables. But the authors reformulation also does not seem to guarantee speedy implementation of the model since the reformulation requires extra continuous variables and constraints that tend to increase linearly as the value of p increases. In an attempt to overcome the limitations of the existing p-median approaches for solving GT cell formation problem, Won and Lee [163] proposed new scheme for implementing the p-median model. They reformulate the p-median model with the binary variables corresponding to the machines not in the candidate set of median machines being excluded from the beginning. This resulted a modified p-median model with reduced number of binary variables.

Shtub [133] modeled the cell formation problem as a generalized assignment problem (GAP) and proved that the model is equivalent to the general formulation of the GT problem (p-median model) and the generalized GT problem (the generalized p-median model). Srinivasan *et al.* [141] showed that the assignment model can overcome some of the limitations of the p-median model, i.e., the number of parts families is not specified a priory. Boctor [16] proposed analytical model that simultaneously cluster or assigns machines and parts to cells. The objective is to minimizing the number of exceptional elements in the final solution. The generalized p-median model in [80] considered alternative routings to increase the number

of cells can be formed to minimize intercell movements. However, this model did not include the cost and capacity of machines. Choobineh [35] developed a similarity measure which uses the manufacturing operations and their sequences and a linear programming model that takes the cost and capacity of machines. However, their formulation did not include different alternative routings explicitly. Rajamani *et al.* [124] proposed three integer programming models that incorporate both budget and machine capacity, as well as alternative process plans. Recent development of mathematical programming approach that used Benders' decomposition for optimal solution can be found in [53, 28]. Wang [155] proposed a new linear assignment algorithm for machine-cell and part-family formation for the design of cellular manufacturing systems. The approach begins with the determination of part-family or machine-cell representatives by means of comparing similarity coefficients between parts or machines and finding a set of the least similar parts or machines. Using the group representatives and associated similarity coefficients, a linear assignment model is formulated for solving the formation problem by allocating the remaining parts or machines and maximizing a similarity index. Based on the formulated linear assignment model, a group formation algorithm was developed.

2.3. Manufacturing Attributes Considered in CMS Design Methods

The design of CM can be broadly divided into two major phases [19]. The first phase of CM design is cell formation. The effectiveness of this phase depends heavily on the consideration of several factors. Review of recently published articles showed that these factors include dynamic cell reconfiguration, lot splitting, sequence of

operations, alternate part routings, operation time and cost, cost of subcontracting part processing, machine capacity, setup cost, tool consumption, workload balancing, cell size limit, machines separation constraints, among others. Detailed list of these attributes is given in Table 2.1. Table 2.2 shows a sample of 18 recently published articles and the corresponding sets of attributes that have been used in their procedures. The attributes used in the model proposed in Chapter 3 are also indicated. As it can be seen from this table the proposed model provides a greater coverage of the attribute than the sample articles. This model is also further expanded in Chapter 6 to include production planning decisions in MRP environment. The impact of lot size on product quality is also taken into consideration. Articles relevant to production planning and the impact of lot size on product quality and are briefly reviewed in Chapter 6.

The second phase of CM design consists of the system design of each of the individual cells. Typical decisions in this phase include equipment layout, design/selection of material handling equipment, assignment of operators to machines, machine loading and scheduling, job dispatching, among others. This research aims at a comprehensive study on the first phase of designing cellular manufacturing systems using a mathematical programming approach.

Table 2.1
List of manufacturing attributes

1. Alternative Routing
   (a) Selecting the Best Route
   (b) Allowing Alternative Routing Coexist
2. Demand Fluctuation
   (a) Deterministic
   (b) Probabilistic
3. Dynamic Cell Reconfiguration
4. Workload Balancing
   (a) Inter-cell Workload
   (b) Intra-cell Workload
5. Lot-Splitting
6. Types of Tools Required by a Part
7. Types of Tools Available on a Machine
8. Machine Proximity Constraint
   (a) Separation Constraint
   (b) Collocation Constraint
9. Sequence of Operation
   (a) Used as input for determine magnitude of material flow
   (b) Used as Similarity measure between parts
10. Setup Cost/Time
    (a) Setup Cost
    (b) Setup Time
11. Cell / Part Family Size Constraint
    (a) Cell Size Constraint
    (b) Part Family Size Constraint
12. Movement of Parts (Material Handling Cost)
    (a) Inter-Cell Movement
    (b) Intra-Cell Movement
13. Facility layout
    (a) Inter-cell Layout
    (b) Intra-cell Layout
14. Operator Allocation
15. Machine Capacity
16. Identical Machines
    (a) Within a Cell
    (b) In the Entire System
17. Machine Investment Cost
18. Subcontracting Cost
19. Tool Consumption Cost
20. Unit Operation Time
21. Operation Cost

Table 2.2
Attributes used in the present study and in a sample of recently published articles

Article/Attributes	1 a	1 b	2 a	2 b	3	4 a	4 b	5	6	7	8 a	8 b	9 a	9 b	10 a	10 b	11 a	11 b	12 a	12 b	13 a	13 b	14	15	16 a	16 b	17	18	19	20	21
Math Model-A (Chapter 3)		×	×		×	×		×	×	×	×	×	×		×		×		×					×	×	×	×	×	×	×	×
Cao and Chen [20]																	×		×					×	×	×				×	
Solimanpur *et al.* [137]	×													×				×	×					×	×	×	×			×	×
Xambre and Vilarinho [165]													×				×		×					×	×	×				×	
Asokan *et al.* [8]							×						×						×	×				×	×	×				×	
Baykasoglu *et al.* [14]	×					×			×	×				×			×	×	×					×	×	×				×	
Diaz *et al.* [41]	×										×	×	×				×		×	×											
Onwubolu and Mutingi [113]						×											×		×												
Akturk and Turkcan [2]	×												×				×		×	×		×		×	×	×	×			×	×
Caux *et al.* [22]	×												×				×		×					×	×	×				×	
Mungwattana [107]	×		×	×	×								×				×		×					×	×	×	×			×	×
Caux *et al.* [22]	×												×				×		×					×	×	×				×	
Zhao and Zhiming [170]						×	×						×				×		×	×				×	×	×				×	
Sofianopoulou [136]	×										×	×	×						×							×					
Wicks and Reasor [160]			×		×								×				×	×	×					×	×	×	×			×	
Chen [29]			×		×												×		×							×	×				
Heragu and Chen [53]		×									×	×					×		×					×	×	×	×			×	
Selim *et al.* [131]									×	×			×			×		×	×		×		×	×	×	×		×	×	×	
Su and Hsu [143]						×	×						×			×	×		×	×	×			×	×	×	×			×	

Note: Attributes' names are referred in Table 2.1

Chapter 3

Mathematical Model-A

3.1. Introduction

The design of cellular manufacturing systems (CMS) involves many structural and operational issues. One of the first important design steps is the formation of part families and machine cells. The effectiveness of this design step heavily depends on the proper consideration of relevant aspects. To this end, a model that incorporates various pragmatic issues is essential. In this chapter a comprehensive mathematical model for the design of cellular manufacturing systems is proposed. The model is formulated based on the tooling requirements of the parts and the tooling available on the machines. It incorporates dynamic cell configuration, alternative routings, lot splitting, sequence of operations, multiple units of identical machines, machine capacity, workload balancing among cells, operation cost, cost of subcontracting part processing, tool consumption cost, setup cost and other practical constraints. A numerical example is presented to demonstrate the model and its potential benefits.

3.2. The Proposed Mathematical Model

This section presents the problem description, the proposed mathematical programming model and manufacturing system features incorporated in the model.

3.2.1. Problem Definition

Consider a manufacturing system consisting of a number of machines to process different parts. Each machine has a number of tools available on it and a part may require some or all of the tools on a given machine. A part may require several operations in a given sequence. An operation of a part can be processed by a machine if the required tool is available on that machine. If the tool is available on more than one machine type then the machines are considered as alternative routings for processing the part. An entire lot of a part may be split into different cells for the processing of an operation if economic advantage can be gained. In addition, we consider the manufacturing system in a number of time periods t, where $t = 1, 2, ..., T$ with $T > 1$. One time period could be a month, a season, or a year. Each machine has a limited capacity expressed in hours during each time period. Machines can have one or more identical copies to meet capacity requirements and reduce/eliminate inter-cell movement. Assuming that the demand for the parts to be processed by the machines vary with t in a deterministic manner, machines are to be grouped into relatively independent cells with minimum inter-cell movement of the parts. In grouping the machines, it is also required that the workload of the cells should be balanced. Machines that cannot be located in the same cell should be separated. Machines that cannot be separated should be co-located. To address this multiple time period cell clustering problem, a mixed integer programming model is formulated. The objective of the model is to minimize machine maintenance and

overhead cost, machine procurement cost, inter-cell travel cost, machine operation and setup cost, tool consumption cost and machine configuration cost for the entire planning time horizon T. The notations used in the model are presented below.

Indices:

Time period index: $t = 1, 2, ..., T$

Part type index: $i = 1, 2, ..., I$

Index of operations of part i: $j = 1, 2, ..., J_i$

Machine index: $k = 1, 2, ..., K$

Tool index: $a = 1, 2, ..., G$

Cell index: $l = 1, 2, ..., L$

Input Data:

$d_i(t)$ - Demand for part i in time period t

V_i - Unit cost to move part i between cells

B_i - Batch size of product i

Φ_i - Cost of subcontracting part i

λ_{jia} - Equals to 1 if operation j of part i requires tool a; 0, otherwise

δ_{ak} - Equals to 1 if tool a is available on machine k; 0, otherwise

h_{jik} - Processing time of operation j of part i on machine type k

w_{jik} - Tool consumption cost of operation j of each part i on machine type k

μ_{jik} - Setup cost for operation j of part i on machine type k

$Q_k(t)$ - Maximum number of machine type k that can be procured at the beginning of period t

$P_k(t)$ - Procurement cost of machine type k at the beginning of period t

G_k - Maintenance and overhead costs of machine type k per time period t

O_k - Operation cost per hour of machine type k

C_k - Capacity of one machine of type k for one time period

LB_l - Minimum number of machines in cell l

UB_l - Maximum number of machines in cell l

R_k^+ - Cost of installing one machine of type k

R_k^- - Cost of removing one machine of type k

q - $0 \leq q < 1$; A factor for the work load of a cell being as low as $q \times 100\%$ from the average work load per cell

$Z_i(t)$ - The number of cells among which an entire lot of part i may split into during time period t for the processing of certain operations; $Z_i(t) \in \{1,\ 2, ...,\ L\}$

M - A large positive number

Θ - A set of machine pairs $\{(k^a, k^b)/k^a, k^b \in \{1, \cdots, K\}, k^a \neq k^b,$ and k^a cannot be placed in the same cell with $k^b\}$

Ω - A set of machine pairs $\{(k^c, k^d)/k^c, k^d \in \{1, \cdots, K\}, k^c \neq k^d,$ and k^c should be placed in the same cell with $k^d\}$

Decision Variables:

General Integer:-

$N_{kl}(t)$ - Number of type k machines to assign to cell l at the beginning of period t

$y_{kl}^+(t)$ - Number of type k machines to add to cell l at the beginning of period t

$y_{kl}^-(t)$ - Number of type k machines to remove from cell l at the beginning of period t

Continuous:-

$\eta_{jikl}(t)$ - The proportion of the total demand of part i with the j^{th} operation

to perform by machine type k in cell l during period t

$\bar{\eta}_i(t)$ - The proportion of the total demand of part i to be subcontracted in time period t

Auxiliary Binary Variables:

The auxiliary binary variables are used to formulate logical constraints. The values of these variables are not required to make decisions for system configuration and operation assignments. These variables are:

$r_{kl}(t)$ - Equals to 1, if type k machines are to be assigned to cell l during time period t; 0, otherwise

$p_{jil}(t)$ - Equals to 1, if operation j of part i is to be processed in cell l during period t; 0, otherwise

3.2.2. Objective Function and Constraints

Following the problem description and notations given in Section 3.2.1, the comprehensive mixed integer programming model for cellular manufacturing system design is presented below.

Objective:

$$\begin{aligned}
\text{Minimize } Z &= \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K} N_{kl}(t)\cdot G_k \\
&+ \sum_{t=1}^{T}\sum_{k=1}^{K} P_k(t)\cdot\left(\sum_{l=1}^{L} N_{kl}(t) - \sum_{l=1}^{L} N_{kl}(t-1)\right) \\
&+ \frac{1}{2}\sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{i=1}^{P}\sum_{j=1}^{J_i-1}\left(d_i(t)\cdot V_i\left|\sum_{k=1}^{K}\eta_{j+1,ikl}(t) - \sum_{k=1}^{K}\eta_{jikl}(t)\right|\right) \\
&+ \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{i=1}^{P}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{jikl}(t)\cdot h_{jik}(t)\cdot O_k
\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{i=1}^{P}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{jikl}(t)\cdot w_{jik} \\
&+ \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{i=1}^{P}\sum_{j=1}^{J_i} \frac{d_i(t)\cdot\eta_{jikl}(t)}{B_i}\cdot\mu_{jik} \\
&+ \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K}\left(R_k^{+}\cdot y_{kl}^{+}(t) + R_k^{-}\cdot y_{kl}^{-}(t)\right) \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{P}\Phi_i\cdot d_i(t)\cdot\bar{\eta}_i(t) \qquad (3.1)
\end{aligned}
$$

Subject to:

$$d_i(t)\cdot\sum_{l=1}^{L}\sum_{k=1}^{K}\eta_{jikl}(t) = d_i(t)(1-\bar{\eta}_i(t)) \quad ; \quad \forall(i,j,t) \qquad (3.2)$$

$$\eta_{jikl}(t) \le \lambda_{jia}\times\delta_{ak} \quad ; \quad \forall(i,j,k,l,t,a) \qquad (3.3)$$

$$\sum_{k=1}^{K}\eta_{jikl}(t) \le p_{jil}(t) \quad ; \quad \forall(i,j,l,t) \qquad (3.4)$$

$$\sum_{l=1}^{L}p_{jil}(t) \le Z_i(t) \quad ; \quad \forall(i,j,t) \qquad (3.5)$$

$$C_k\cdot N_{kl}(t) - \sum_{i=1}^{P}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{jikl}(t)\cdot h_{jik} \ge 0 \quad ; \quad \forall(k,l,t) \qquad (3.6)$$

$$\sum_{k=1}^{K}\sum_{i=1}^{P}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{jikl}(t)\cdot h_{jik} \ge \frac{q}{L}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{i=1}^{P}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{jikl}(t)\cdot h_{jik} \quad ; \quad \forall(l,t) \qquad (3.7)$$

$$\sum_{l=1}^{L}N_{kl}(t) - \sum_{l=1}^{L}N_{kl}(t-1) \ge 0 \quad ; \quad \forall(k,t) \qquad (3.8)$$

$$LB_l \le \sum_{k=1}^{K}N_{kl}(t) \le UB_l \quad ; \quad \forall(l,t) \qquad (3.9)$$

$$N_{kl}(t) = N_{kl}(t-1) + y_{kl}^{+}(t) - y_{kl}^{-}(t) \quad ; \quad \forall(k,l,t) \qquad (3.10)$$

$$N_{kl}(t) \le M\cdot r_{kl}(t) \quad ; \quad \forall(k,l,t) \qquad (3.11)$$

$$r_{kl}(t) \le N_{kl}(t) \quad ; \quad \forall(k,l,t) \qquad (3.12)$$

$$r_{k^al}(t) + r_{k^bl}(t) \le 1 \quad ; \quad (k^a,k^b)\in\Theta,\ \forall(l,t) \qquad (3.13)$$

$$r_{k^cl}(t) - r_{k^dl}(t) = 0 \quad ; \quad (k^c,k^d)\in\Omega,\ \forall(l,t) \qquad (3.14)$$

$$0 \leq \bar{\eta}_i(t) \leq 1 \quad ; \quad \forall (i,t) \tag{3.15}$$

$$y_{kl}^{+}(t),\ y_{kl}^{-}(t),\ N_{kl}(t) \in \{0,1,2,\cdots\} \quad ; \quad \forall (k,l,t) \tag{3.16}$$

$$p_{jil}(t) \text{ and } r_{kl}(t) \in \{0,1\} \quad ; \quad \forall (i,j,k,l,t) \tag{3.17}$$

Model Objective Function: The 1st term of Z is machine maintenance and overhead costs. The 2nd term is machine procurement cost where $\sum_{l=1}^{L} N_{kl}(t),\ \forall t \geq 1$, is the number of machines of type k in the system at the beginning of period t. $\sum_{l=1}^{L} N_{kl}(0)$ is the number of machines of type k available from a previous system if the problem is to reconfigure an existing system. For setting up a new system, $\sum_{l=1}^{L} N_{kl}(0) = 0,\ \forall k$. The 3rd term represents the inter-cell material handling cost. Assume that the costs of moving the same material between different cells are the same since the fixed costs involved in moving materials are normally large while the distance related cost components are typically small [53], and hence are negligible. The 4th-7th terms of Z are machine operating cost, tool consumption cost, setup cost, and machine relocation cost, respectively. The 8th term is the cost for subcontracting parts.

Model Constraints: Eq. (3.2) is to ensure that if a part is not subcontracted, the processing of each operation of this part must be assigned to a machine. An assignment of an operation of a part is permitted only to a machine having the required tool using (3.3). This constraint is also for limiting the values of $\eta_{jikl}(t)$ within $[0,1]$. The processing of an operation j of part i is allowed to be performed in at most $Z_i(t)$ cells in time period t with (3.4) and (3.5). Machine capacity constraints are in (3.6). Workload balancing among cells is enforced with (3.7) where the factor $q \in [0,1)$ is used to determine the extent of the workload balance. If the number of cells is L, the minimum allowable workload of a cell is $\frac{q}{L} \times 100\%$ of the total

workload in terms of processing time. The maximum allowable workload is given by $\left(\frac{q}{L}+1-q\right) \times 100\%$ of the total workload. If q is chosen close to 1.0, the allowable workload of each cell will be close to the average workload given by $\frac{1}{L} \times 100\%$ of the total workload. Eq. (3.8) implies that the number of type k machines used any period is greater than or equal to that of the previous period. This means that the model is not going to remove extra machines of any type if that type of machines happen to be in excess in a certain time period. The presence of extra machines in the system increases system flexibility and reliability by providing alternative routes during machine breakdown. Lower and upper bounds on the sizes of the cells are enforced with (3.9). Eq. (3.10) is to ensure that the number of machines of type k in the current period in a particular cell is equal to the number of machines in the previous period, adding the number of machines moved in and subtracting the number of machines moved out of the cell. Eqs. (3.11) and (3.12) are for setting $r_{kl}(t)$ to 1 if at least one type k machine is located in cell l during period t, 0 otherwise. Eq. (3.13) is to ensure that machine pairs included in Θ should not be placed in the same cell. Eq. (3.14) is to ensure that machine pairs included in Ω should be placed in the same cell. The values of $\bar{\eta}_i(t)$ are limited within $[0,1]$ by (3.15). Eq. (3.16) and (3.17) are integrality constraints.

3.2.3. Features of the Model

The distinguishing feature of the model presented in this chapter is that it simultaneously addresses several pragmatic issues in the design of cellular manufacturing systems. Some of these issues are briefly discussed below.

Dynamic Reconfiguration of Cells: In the presence of product mix variations, cell reconfiguration is an obligatory issue that should not be overlooked if the

manufacturing system is to remain efficient. With increased demand for manufacturing flexibility, this problem becomes more prominent in designing manufacturing cells [29]. Most developed procedures use static product mix to cluster machines into cells. In a dynamic manufacturing environment, it is very likely that production demand and part mix change with time. System reconfiguration is required in order to efficiently operate the system under such product mix variations. The issue of designing CMS in a dynamic environment was discussed in [107], [130], [160], [29] and [52]. In this research, we attempt to address this issue in a more realistic way by incorporating many other practical matters into the proposed mathematical model.

Alternative Routings: The presence of alternative routings is very customary in many discrete, multi-batch, small lot size production environments. Routing flexibility increases the number of ways that manufacturing cells can be formed to save cost. The proposed mathematical model has been formulated based on the tooling requirements of the parts and tooling availability on the machines. If a tool is available on more than one machine, then these machines can make alternative routings for operations requiring that particular tool. The previous research work found in the literature considering dynamic cell reconfiguration has either ignored routing flexibility [29] or only found the best route for each part from the available routes and disregarded the remaining routings from further consideration [160, 107]. However, ignoring the remaining alternative routings can result in one or more of the following undesirable effects: machine under utilization, increased operation cost and additional investment on machines. For this reason, in the model proposed in this chapter, alternative routings are allowed to coexist and share the total production volume if economic advantage can occur.

Lot Splitting: Lot splitting is a process used primarily in batch manufacturing scheduling. It divides large orders into smaller batches providing the opportunity

for simultaneous processing of orders on more than one work center. This may result in reduced flow time [63] and better due date performance [154]. In the context of cellular manufacturing systems operation, [91] and [144] used the concept of lot splitting in order to improve the effectiveness of scheduling decisions. However, some lot splitting decisions that have not been considered at the early stage of the system design may not be optimal in terms of intercellular movement, operation cost and other cell formation objectives. In our research, we introduce lot splitting at the design phase of cellular manufacturing systems as it may result in improved machine utilization, reduced inter-cell movement, decreased operation cost, reduced machine investment and evenly distributed workload. If a cell designer sets the value of $Z_i(t)$ greater than one in Eq. (3.5), the whole lot may split into $Z_i(t)$ cells allowing simultaneous processing of an operation of a part provided that the required machines are available in these cells. The lot splitting introduced during this design phase can be utilized in the scheduling function with no or insignificant negative impact on cell formation decisions.

Sequence of Operations: The consideration of sequence of operations helps to determine the exact count of inter-cell movement of parts. Models that do not use operation sequence data may result in non-optimal solutions in terms of inter-cell movement of parts. In our mathematical model, operation sequence data is used in order to more accurately calculate inter-cell movement.

Workload Balancing: Workload balancing contributes to a smooth running of the system and better performance in terms of throughput, makespan, flow time and tardiness [75]. Balancing workload reduces work-in-process and improves the flow of the parts through the system. A constraint enforcing workload balancing among cells (Eq. 3.7) is included in the proposed model. In this constraint, the factor $q \in [0, 1)$ determines the extent of the workload balance. If the number of

cells is L, the minimum allowable workload of a cell is $\frac{q}{L} \times 100\%$ of the total workload in terms of processing time. This corresponds to the maximum allowable workload given by $\left(\frac{q}{L} + 1 - q\right) \times 100\%$ of the total workload.

Machine Adjacency Constraint: A number of authors addressed machine adjacency requirement in CMS design [41, 117, 136, 53]. Such requirements exist since some machines must be separated from each other while other machines must be placed together due to technical and safety considerations. For example, machines that produce vibrations, dust, noise, or high temperatures may need to be separated from electronic assembly and final testing. In other situations, certain machines should be placed in the same cells. For example, a heat treatment station and a forging station may be placed adjacent to each other for safety reasons. Machines that share a common resource or those that require a particular operator's skill may also be placed in a same cell.

Other Features: The proposed model also incorporates cell size limits, setup cost, tool consumption cost and subcontracting cost of parts processing.

3.2.4. Linearizing the Model

The proposed model is a non-linear model because of the absolute value in the third term of the objective function. This term can be linearized by introducing non-negative real variables $n^+_{ijl}(t)$ and $n^-_{ijl}(t)$ and a binary variable $\beta_{ijl}(t)$ as follows:

The term $\left|\sum_{k=1}^{K} \eta_{i,j+1,kl}(t) - \sum_{k=1}^{K} \eta_{jikl}(t)\right|$ in the third cost element of the objective function is replaced by $n^+_{ijl}(t) + n^-_{ijl}(t)$ with the added constriants:

$$\sum_{k=1}^{K} \eta_{i,j+1,kl}(t) = \sum_{k=1}^{K} \eta_{jikl}(t) + n^+_{ijl}(t) - n^-_{ijl}(t) \tag{3.18}$$

$$n^+_{ijl}(t) \leq M^{\infty} \cdot \beta_{ijl}(t) \tag{3.19}$$

$$n^{-}_{ijl}(t) \leq M^{\infty} \cdot (1 - \beta_{ijl}(t)) \tag{3.20}$$

$$\beta_{ijl}(t) \text{ is binary.} \tag{3.21}$$

After this term is linearized, the integer programming model has objective function with linear terms only and all constraint functions are linear functions. This linearization is required in order to solve the model using commercially available software for small size problem.

3.3. Numerical Examples

Several example problems, all solved with LINGO, a commercially available optimization software, are presented in this section. Example 1 is explained in detail for its input data and computational results. Since other example problems are similar to Example 1, only summarized results are presented to further illustrate the CMS design issues addressed with the proposed model.

3.3.1. Example 1

In solving this example, we consider 10 different types of machines, 25 part types, and two planning time periods. The machines are to be grouped into three relatively independent cells and reconfiguration is to be performed at the beginning of the second period to respond to the changes of production demand. For this numerical example, the model presented in Section 3.2 has a total of 6,438 variables having potentially non-zero values. 2,100 of these variables were integers. The corresponding number of functional constraints is 6,723.

Input Data

The input data of this example problem are given in Tables A.1 to A.6. These sets of data were generated randomly within the ranges of data found in most published articles and case studies. Table A.1 contains data on production batch size, unit cost of inter-cell movement and the demand for the parts in the two time periods. Table A.2 shows the tools required by the operations to process the parts. Table A.3 presents the tool availability on the machines. Table A.4 contains data related to alternative routings. The data in Table A.4 are generated by matching tool requirement by operations of the parts and tool availability on the machines. For example, Table A.2 shows the first operation of part 1 requires tool A01. This tool is available on machine types 1 and 2 (Table A.3). Hence machine types 1 and 2 can be used in alternative routings for the first operation of part 1. The corresponding setup costs and operation times are assumed known as shown in Table A.4. Machine cost data and other machine data are given in Table A.5. These include machine overhead costs, operating costs, machine installation and removal costs, machine capacities, the maximum number of machines that can be procured and machine procurement costs. We assume that the problem is to setup a new system from the beginning thus $N_{kl}(0) = 0$, $\forall t$. Table A.6 contains some miscellaneous data. It shows the number of cells to be formed, lower and upper bounds for the cell sizes, list of machine pairs that cannot be placed in a same cell and work load balancing factor q. For this example problem, we assume that no part processing will be contracted out, so the subcontracting cost Φ_i was given a large number for each part type. We also assume that the variations in tool consumption cost for various alternative routings are negligible because of the smaller number of parts considered in the example.

Solution of the Example Problem

Using the model proposed in Section 3.2 and the above data, optimal cell formation and part processing decisions are found using LINGO software. The cells generated during each time period and the part assignment to the various cells are given in Tables A.7 and A.8. The assignment of the various operations to the machines is indicated by the decision variable $\eta_{jikl}(t)$. The number of variables $\eta_{jikl}(t)$, for all i, j, k, l and t is very large. Here we only present a sample set of $\eta_{jikl}(t)$ for part types 1, 6, 10, 15 and 21 shown in Table A.9. As can be seen from this table, part 15 is entirely processed in cell 1 during period 1, with operations 1 and 4 processed by machine type 1, operation 2 processed by machine type 3, operations 3 and 5 processed by machine type 4 and operation 6 by machine type 8. This is indicated in Table A.7 by a unit value corresponding to machine types 1, 3, 4 and 8 in cell 1. Similar to part 15, part 1 is also processed in one cell during period 1. Notice that the fourth operation of part type 1 is performed partially by machine type 3 and partially by machine type 4 due to its alternative routings that coexist. Part 10 is processed partially in cell 1 and partially in cell 2 due to lot splitting. This result is reflected in Table A.7. One can see that part 10 appears in columns 7 and 10 and there are no values outside the diagonal block corresponding to this part. Similar to part 10, part 6 is also processed in two cells during period 2. Notice that there is a combined effect of lot splitting and alternative routings as the fourth and fifth operations of this part are performed by machine type 1 in cell 2 and by machine type 2 in cell 3. The first four operations of part 21 are processed partially in cell 2 and partially in cell 3. The fifth operation is processed partially in cell 3 and partially in cell 1 while the last three operations are performed within cell 1. Hence, there is an inter-cell movement from cell 2 and cell 3 to cell 1. This is reflected by

elements outside the diagonal block in Table A.8.

The reconfiguration performed at the beginning of period 2 is indicated by the decision variables $y_{kl}^{+}(t)$ and $y_{kl}^{-}(t)$ and the values of these variable can be determined from the data given in Tables A.7 and A.8. For example, six units of machine type 2, one unit of machine type 3 and four units of machine type 6 are added to cell 1 in period 2. At the same time, one unit of each of machine types 1, 4, 7, 8, 9, 10 and two units of machine type 5 are removed from cell 1.

Workload Distribution

The above discussed solution was based on the workload balancing factor q set to 0.9. Hence, for the three cells generated, the minimum allowable work load is $\frac{0.9}{3} \times 100 = 30\%$ of the total workload in processing time with the maximum being $\left(\frac{0.9}{3} + 1 - 0.9\right) \times 100 = 40\%$. In order to see the impact of the constraint that enforces workload balancing, we also run the example problem with $q = 0$. The resulting workload distributions and the corresponding objective function values for these two different values of q are given in Table 3.1.

Table 3.1: Workload distribution

		Workload of cells in processing time and as a percentage of the total workload				Objective function
q	Cell	Period-1		Period-2		value
	1	1527821	40%	2053727	44%	
0	2	1380987	36%	1597487	34%	2,613,864.00
	3	957854	25%	1025005	22%	
	1	1259986	33%	1644312	35%	
0.9	2	1160890	30%	1616507	35%	2,626,995.00
	3	1448759	37%	1397494	30%	

As it can be seen from Table 3.1, for $q = 0$ (or without the constraint that enforce workload balancing), there are significant workload differences among the cells.

In period 1, the workload difference between cell 1 and cell 2 is 569967 processing time in minutes. If we assume an average processing time of the operations to be 12 minutes, then 47498 more operations are performed in cell 1 than in cell 3. In period 2, cell 3 receives only half of the load of cell 1. Such unbalanced workload may lead to a poor performance of the system in terms of production throughput and increased work-in-process. For $q = 0.9$, the workload is evenly distributed among the cells with certain increment of the objective function value.

Cost Savings

We briefly discuss cost saving issue related to the example problem from different aspects of the proposed model. These include dynamic cell reconfiguration, lot splitting and routing flexibility. To investigate the cost saving as a result of these features, we solve the model presented in Section 3.2 by eliminating these features one at a time. If we add the constraint:

$$y_{kl}^{-}(t) = y_{kl}^{+}(t) = 0, t \geq 2 \tag{3.22}$$

to the basic model, it will enforce that all required machines be installed at the beginning of period 1 and there will be no system reconfiguration. If $Z_i(t)$ is set equal to 1 for all i and t in Eq. (3.5) of the basic model, then no lot-splitting can take place. If we add the constraint:

$$\sum_{t=1}^{T}\sum_{l=1}^{L}\eta_{1,1,1,l}(t) = 0 \tag{3.23}$$

to the model, it prevents the use of machine type 1 for the first operation of part 1 since this machine involves a higher sum of setup and operation cost per unit of part than machine type 2 (i.e., $[\frac{\mu_{1,1,1}}{B_1} + O_1 \times h_{1,1,1}] > [\frac{\mu_{1,1,2}}{B_1} + O_2 \times h_{1,1,2}]$). Similar constraints can be added corresponding to each alternative routes of a part involving

higher sum of setup and operation cost per unit of that part. This process will eliminate all alternative routings having higher setup and operations cost. Thus each operation will have exactly one route and alternative routes will no longer be used in the cell formation decision. On the other hand, if we add the constraints:

$$\sum_{l=1}^{L} \eta_{1,1,1,l}(t) \leq M^{\infty} \cdot X_1(t) \tag{3.24}$$

$$\sum_{l=1}^{L} \eta_{1,1,2,l}(t) \leq M^{\infty} \cdot (1 - X_1(t)) \tag{3.25}$$

$$X_1(t) \text{ is binary.} \tag{3.26}$$

to the basic model, then the model will select either machine type 1 or machine type 2 for the first operation of part 1. Similar sets of constraints can be imposed corresponding to other operations having alternative routings. This process will inhibit coexistence of alternative routings.

Table 3.2: Cost saving as a result of some features of the model

Feature inhibited from the basic model	Objective Function Value	Cost Saving by Considering the feature
None	2,626,995.00	NA
Dynamic Reconfiguration of cells	2,661,179.00	34,184.00
Lot Splitting	2,696,860.00	69,865.00
Alternative Routings	2,732,050.00	105,055.00
Coexistence of Alternative Routings	2,656,860.00	29,865.00

By eliminating the features mentioned above one at a time from the basic model using the corresponding sets of constraints, we run the example problem to see their impacts on the objective function. As it can be seen from Table 3.2, cost savings are significant for the example problem resulted from dynamic reconfiguration, lot splitting and routing flexibility.

3.3.2. Other Example Problems

We further illustrate the proposed model using ten other numerical examples, problems 2 to 11. The data for these examples were generated by varying the data related to the following problem aspects.

- Number of planning periods and demands for part processing
- Number of cells, number of machines, machine capacities, machine procurement, holding, and operation costs
- Number of part types
- Numbers and sequences of operations of the parts
- Setup costs and processing times of the operations

The variations mentioned above are within relatively small ranges but do not follow any particular pattern. General features of these additional problems are in Table A.10.

A summary of the impact of the workload balancing constraint on the workload distributions and objective function values of these 10 problems is in Table 3.3. In the third and fourth columns of this table are the maximum workload differences, expressed as a percentage of the total workloads. From these columns it can be seen that there are considerable workload differences in these examples if the workload balancing constraint is not imposed. A maximum workload difference of 37% is observed in problem 2, where one of the cells carries 55% and another cell carries only 18% of the total workload. The averages of the maximum workload differences of all the problems are 7.0% and 23.8% with and without the workload balancing constraint, respectively. The last column of this table are the increments of the

objective function value due to workload balancing constraint. As can be seen from this column, the increment of the objective function value is less than 0.1% for the seven of the ten problems and the average percentage increment is 0.14%. In Table 3.4 we present the cost savings observed from these 10 example problems as a result of dynamic reconfiguration, lot splitting, alternative routings and allowing alternative routings to coexist. As can be seen from this table, lot splitting and alternative routing have resulted in significant cost savings with the averages being 5.95% and 6.47%, respectively. Cost saving from lot splitting can be due to reduced inter-cell movement, reduced machine investment, and better machine utilization. It can also enable workload balancing with minimal inter-cell movement since the processing of an operation of a batch can be allocated to different cells. The cost saving from alternative routings can be from reduced inter-cell movement, operation cost, setup costs, and machine investment cost since it can increase the number of ways in which the cells can be formed to reduce these costs. Dynamic reconfiguration and allowing alternative routings to coexist have resulted in cost savings in these 10 problems with the averages being 0.50% and 0.53% respectively. These small percentages can mean a considerable cost saving when large investments are considered.

3.4. Summary and Discussion

In this chapter we consider a multiple time period system planning problem in cellular manufacturing systems. We proposed a comprehensive mathematical model to generate manufacturing cells which may be sustained for a number of planning time periods. The model attempts to minimize machine investment cost, inter-cell material handling cost, operating cost, cost of subcontracting part processing, tool

Table 3.3
Impacts of the workload balancing constraint on the workload distributions and objective function values of the 10 arbitrarily generated problems

Problem No.	Maximum Cell Load Difference as % of Total			Objective Value Increment %
	Period	$q = 0$	$q = 0.9$	
2	1	37	10	0.05
	2	29	8	
3	1	25	3	0.11
	2	24	10	
4	1	33	4	0.03
	2	26	4	
5	1	31	7	0.01
	2	35	10	
6	1	11	6	0.02
	2	29	7	
7	1	33	9	0.04
	2	18	6	
8	1	10	9	0.07
	2	27	8	
	3	9	8	
9	1	13	10	0.33
	2	11	6	
	3	32	10	
10	1	21	7	0.04
	2	15	5	
11	1	29	4	0.74
	2	25	4	
Average	–	23.8	7.0	0.14

Table 3.4
Cost savings resulted from some features of the model on 10 arbitrarily generated problems

	Feature Eliminated-Objective/Cost Saving				
Problem No.	None	Dynamic Reconfiguration	Lot Splitting	Alternative Routing	Coexistence of Alternative Routings
2	3,296,888.00	3,308,990.00	3,661,735.00	3,364,681.00	3,326,555.00
	NA	12,102.00	364,847.00	67,793.00	29,667.00
3	3,794,331.00	3,804,331.00	3,879,594.00	3,848,139.00	3,824,084.00
	NA	10,000.00	85,263.00	53,808.00	29,753.00
4	2,022,009.00	2,024,922.00	2,065,277.00	2,088,094.00	2,023,695.00
	NA	2,913.00	43,268.00	66,085.00	1,686.00
5	2,924,824.00	2,929,882.00	2,986,842.00	3,072,987.00	2,940,365.00
	NA	5,058.00	62,018.00	148,163.00	15,541.00
6	1,529,347.00	1,548,218.00	1,548,005.00	1,666,412.00	1,530,340.00
	NA	18,871.00	18,658.00	137,065.00	993.00
7	1,535,497.00	1,538,605.00	1,549,114.00	1,705,372.00	1,536,560.00
	NA	3,108.00	13,617.00	169,875.00	1,063.00
8	2,067,705.00	2,070,649.00	2,141,568.00	2,283,562.00	2,069,057.00
	NA	2,944.00	73,863.00	215,857.00	1,352.00
9	2,214,939.00	2,227,884.00	2,556,638.00	2,246,319.00	2,231,734.00
	NA	12,945.00	341,699.00	31,380.00	16,795.00
10	1,744,313.00	1,757,518.00	1,844,063.00	1,769,873.00	1,763,632.00
	NA	13,205.00	99,750.00	25,560.00	19,319.00
11	2,439,456.00	2,448,741.00	2,511,847.00	2,478,229.00	2,449,698.00
	NA	9,285.00	72,391.00	38,773.00	10,242.00
Average Saving in Percent		0.50	5.95	6.47	0.53

consumption cost, setup cost and system reconfiguration cost in an integrated manner. The model also addresses many pragmatic issues such as alternative routings, lot splitting, sequence of operations, workload balancing and separation requirement on machines that should not be located in a same cell due to safety, vibration and other considerations. For small size problems, commercially available integer programming codes may be used to solve the formulation. Computational experience on such small problems showed that a significant amount of cost savings can be achieved by considering system reconfigurations, lot splitting and system flexibility. Our computational results also showed that there are significant differences on workload distribution among the cells, if workload balancing is not attempted.

Chapter 4

GA Based Solution Procedure for Math Model-A

4.1. Introduction

In the previous chapter, we present a comprehensive mathematical programming model for the formation of manufacturing cells over multiple time periods. For small size problems, off-the-shelf optimization software LINGO was used to solve this formulation. For solving large size problems, branch and bound based general search algorithm employed by such software cannot give optimal or near optimal solutions within acceptable computational times on widely available platforms such as a PC computer. To this end, in this chapter, we develop an efficient heuristic based on genetic algorithm (GA). GAs are efficient search methods based on the principles of natural selection and genetics (Holland [55]). They are being applied successfully to find acceptable solutions to a variety of problems in different disciplines. GAs are generally able to find good solutions in reasonable amounts of time. The performance of the proposed GA was evaluated against LINGO and the

result was very encouraging. To further improve the performance of the developed GA, we considered parallel implementation of the algorithm on cluster of workstations. Numerical examples showed that the parallel implementation demonstrates a remarkable improvement of the search performance in terms of both computation time and solution quality. To our knowledge the use of parallel GAs (PGA) in CMS design was reported only in [12], though a number of researchers agreed that CMS design is a complex problem. As yet, the PGA methodology has not been exploited for the design of CMSs and so this novel approach is attempted in this chapter.

4.2. Components of the Proposed GA

The genetic algorithm developed in this work is tailored in accordance with the nature of the cell formation problem. This includes the development of solution representation, fitness evaluation, genetic operators and heuristic techniques that are specific to the model presented in the previous chapter.

4.2.1. The Chromosomal Encoding of a Solution

The chromosomal encoding of a solution is the first task in applying a genetic algorithm. In this research we developed a chromosomal representation of a solution which can satisfy some of the constraints of the model. Fig. 4.1 illustrates a chromosome structure for a particular problem with two planning period and 11 part types. The variable $\bar{\eta}_i(t)$ takes a value in [0, 1] denoting the proposition of the total demand of part i subcontracted during period t. The variable c_j takes a value in $\{1, 2, ...L\}$ representing the cell in which operation j is performed. The x_j's assume values in [0, 1] and are used to calculate the proportions of the production volume among alternative routings. Operations without alternative routings do not

have x's associated with them. In the chromosome structure, parts which do not have demand in a given period are not included in the segment of the chromosome representing the product mix for that particular period. For example parts 2, 5 and 10 shown in Fig. 4.1 do not appear in the first half of the chromosome since there is no such demand in period 1. The detailed representation of part 6 is shown in this figure. This part is assumed to have nine operations and the 2^{nd}, 8^{th} and 9^{th} operations have two alternative routings each. The 4^{th} operation has three alternative routings and the remaining operations have only one route each. Lot splitting is not taken into consideration in this solution representation and hence each operation will be performed in a single cell. Lot splitting has been addressed in Chapters 6 and 7.

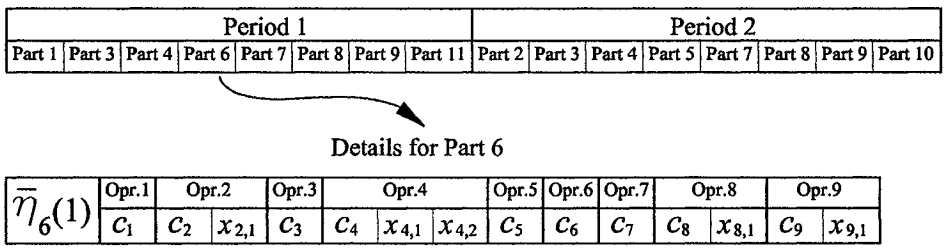


Figure 4.1: A chromosome structure for a two-planning period problem

4.2.2. Decoding a Chromosome

The decision variables $\bar{\eta}_i(t)$ and $\eta_{jikl}(t)$ are determined by decoding a chromosome under consideration. The value of $\bar{\eta}_i(t)$ is directly read from the chromosome. For an operation with n alternative routings along machines $k^1, k^2, \cdots, k^n$, the values of $\eta_{jik^1l}(t), \eta_{jik^2l}(t), \cdots \eta_{jik^nl}(t)$ are determined using the sets of equations given in Fig. 4.2. In this set of equations the values of x_{j1}, x_{j2}, $\cdots, x_{j,n-1}$ are

obtained from the chromosome and x_{jn} is set to 1. The subscript l takes the value c_j which is also obtained from the chromosome.

$$
\begin{aligned}
\eta_{jik^1l}(t) &= (1-\bar{\eta}_i(t)) \times x_{j1}, \\
\eta_{jik^2l}(t) &= (1-\bar{\eta}_i(t)) \times (1-x_{j1}) \times x_{j2}, \\
\eta_{jik^3l}(t) &= (1-\bar{\eta}_i(t)) \times (1-x_{j1}) \times (1-x_{j2}) \times x_{j3}, \\
&\vdots \quad\quad \vdots \\
\eta_{jik^nl}(t) &= (1-\bar{\eta}_i(t)) \times (1-x_{j1}) \times (1-x_{j2}) \times \cdots \times (1-x_{j,n-2}) \times x_{j,n-1} \times x_{jn}, \\
&\& \\
\eta_{jikl}(t) &= 0\ ; \quad k \notin \{k^1, k^2, \cdots k^{n-1}\},\ l \neq c_j
\end{aligned}
$$

Figure 4.2: Equations required for decoding a chromosome

As an example, for an operation with three alternative routings along machines k^1, k^2 and k^3, the values $\eta_{jik^1l}(t)$, $\eta_{jik^2l}(t)$ and $\eta_{jik^3l}(t)$ are determined using the first three equations for $x_{j3} = 1$ while x_{j1} and x_{j2} are obtained from the chromosome. Here, it can be seen that:

$$
\begin{aligned}
\sum_{l=1}^{L}\sum_{k=1}^{K} \eta_{jikl}(t) &= \eta_{jik^1,\,cj}(t) + \eta_{jik^2,\,cj}(t) + \eta_{jik^3,\,cj}(t) \\
&= (1-\bar{\eta}_i(t)) \times x_{j1} + (1-\bar{\eta}_i(t)) \times (1-x_{j1}) \times x_{j2} \\
&+ (1-\bar{\eta}_i(t)) \times (1-x_{j1}) \times (1-x_{j2}) \\
&= 1-\bar{\eta}_i(t)
\end{aligned}
$$

This ensures the constraint in Eq. (3.2). From Fig. 4.2, $\eta_{jikl}(t) = 0$ for all $k \notin \{k^1, k^2, k^3\}$. This ensures the constraint in Eq. (3.3) which restricts the assignment of an operation to those machines that can process the operation. Moreover, $\eta_{jikl}(t) = 0$ for $l \neq c_j$. This is in agreement with the constraints in Eqs. (3.4) and (3.5) which limit the assignment of an operation to only one cell. From the above discussion it can be seen that the chromosomal encoding enables a randomly generated solution satisfying the constraints in Eqs. (3.2)–(3.5) of the model presented

in Chapter 3

4.2.3. The Fitness Function

The purpose of the fitness function is to measure the fitness of candidate solutions in the population with respect to the objective and constraint functions of the model. It is given by Eq. (4.2.3) as the sum of the objective function and the penalty terms of constraint violations. Workload balancing, cell size constraints and machine co-location are enforced by such penalty terms. The factors f_{wb}, f_{cs} and f_{mc} are used for scaling the penalty terms corresponding to these constraints. The minimum allowable and the actual work performed in a cell in time period t are $W_{min}(t)$ and $W_l(t)$ and computed by Eqs. (4.2) and (4.3), respectively.

$$\begin{aligned} E = \text{Model Objective Function} \;+\; f_{wb} \cdot \sum_{t=1}^{T}\sum_{l=1}^{L} \max\{0,\; W_{min}(t) - W_l(t)\} \\ +\; f_{cs} \cdot \sum_{t=1}^{T}\sum_{l=1}^{L} \max\left\{0,\; \sum_{k=1}^{K} N_{kl}(t) - LB_l\;,\; \sum_{k=1}^{K} N_{kl}(t) - UB_l\right\} \\ +\; f_{mc} \cdot \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{\forall(k^c,k^d)\in\Omega} |r_{k^c l}(t) - r_{k^b l}(t)| \end{aligned} \tag{4.1}$$

$$W_{min}(t) = \frac{q}{L}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{i=1}^{P}\sum_{j=1}^{J_i} d_i(t) \times \eta_{ijkl}(t) \times h_{ijk} \tag{4.2}$$

$$W_l(t) = \sum_{k=1}^{K}\sum_{i=1}^{P}\sum_{j=1}^{J_i} d_i(t) \times \eta_{ijkl}(t) \times h_{ijk} \tag{4.3}$$

For solutions of a given generation, the most promising ones are those with the minimum E. However, genetic algorithm works with maximization function. Hence, the raw fitness scores need to be transformed so that the minimum raw fitness will correspond to the maximum transformed fitness. This is achieved using Eq. (4.4)

where E_{max} and E_{min} are the maximum and the minimum values of the raw fitness E in the current population.

$$\widetilde{E} = \begin{cases} 1 & ; \ if \ E_{max} = E_{min} \\ \frac{E_{max}-E}{E_{max}-E_{min}} & ; \ if \ \frac{E_{max}-E}{E_{max}-E_{min}} > 0.1 \\ 0.1 & ; \ otherwise. \end{cases} \tag{4.4}$$

4.2.4. Genetic Operators

Genetic operators make the population to evolve by creating promising candidate solutions to replace those less promising ones. These operators are generally classified as selection, crossover, and mutation operators.

Selection Operator: The selection operator selects, with replacement, individuals having the potential to replace the current population according to the transformed fitness function $\widetilde{F}$. This selection operator may be implemented by simulating a biased roulette wheel where each individual chromosome in the current population has a roulette wheel slot sized in proportion to its transformed fitness [45]. Once the mating pool is formed, individuals are randomly paired and crossover operator will be applied to each pair with certain probability.

Crossover Operators: The crossover operators produce children by exchanging information contained in the parents. The genetic algorithm used to solve the proposed model employs the standard single-point crossover operator. It randomly generates a single crossover point along the length of the chromosome. This crossover point divides each of the parent chromosomes into two segments. The single-point crossover

swamps the right-hand-side segments of the parents. The algorithm also incorporates three problem specific crossover operators: the period-swap, part-swamp and operations-swamp crossover operators. The period-swap crossover operator randomly selects a period in the planning horizon and exchanges the subcontracting and operations assignment information of all the parts for the selected period between the parents. The part-swap (operation-swamp) crossover operator randomly selects a part (an operation) in the length of the chromosome and exchanges all the information about this part (operation) between the parents.

Mutation Operators: Selection and crossover do not introduce new genetic material into the population pool. This task is performed by the mutation operators acting at the gene level to alter information contained in the gene. Mutation operators are critical to the success of genetic algorithms since they diversify search directions and avoid premature convergence to local optima. Hong *et al.* [56] suggested that each problem, even each stage of the genetic process in a single problem, may require appropriately defined multiple mutation operators for best results. In this research, we developed six different mutation operators. These are:

- part-level cell mutator,
- operation-level cell mutator,
- subcontract mutator,
- alternative route mutator,
- subcontract degenerator and
- alternative route degenerator.

The part-level cell mutator performs cell mutation. This operator randomly alters the c_j's of all the operations of a part to other identical values in $\{1, 2, \cdots L\}$.

It is applied during the first phase of the genetic search where the quest is to find the best configuration with independent cells. Similarly, operation-level cell mutator alters the value of c_j's. However, this operator is applied for each operation independently and may result in different values of c_j's of the operations of a part. Hence, it may result in inter-cell movements. For this reason, this operator is applied in the second phase of the genetic search where the attempt is to optimize the cost of inter-cell movement along other cost terms of the model. Moreover, this operator is applied at a lower mutation rate than part-level cell mutator to avoid unnecessary perturbations. Subcontract mutator randomly steps up or down the variable $\bar{\eta}_i$ with a step-amount. This operator also takes care of the constraint in Eq. (3.15) of the model. Alternative route mutator randomly steps up or down the variable x_j's for all the operations having alternative routings with a step-amount while keeping these values in [0, 1]. Since both $\bar{\eta}_i$ and x_j's are kept in [0, 1], the variable η_{jikl} will also be kept in the same interval (see Fig. 4.2) to ensure the constraint in Eq. (3.15) of the model. All the mutation operators discussed above are applied with small probabilities.

The subcontract and alternative route degenerators are non-probabilistic mutation operators. The subcontract degenerator sets the value of $\bar{\eta}_i = 0$ if its current value is less than a degeneration limit, d_1. It also sets $\bar{\eta}_i = 1$ if its current value is greater than $1 - d_1$. The alternative routing degenerator sets a value of $x_j = 0$ or 1 based on the magnitude of flow along the alternative routes of an operation. For an operation j of part i with two alternative routings, the operator sets $x_{j1} = 0$ if its current value time $(1 - \bar{\eta}_i)$ is less than a degeneration limit, d_2. It also sets $x_{j1} = 1$ if its current value times $(1 - \bar{\eta}_i)$ is greater than $1 - d_2$. The purpose of developing and applying these two operators is to speed up convergence by quickly degenerating insignificantly small values of the continuous decision variables.

4.2.5. Repair Heuristic

The Repair Heuristic is run for each individual chromosome representing a solution which violates the machine separation constraint. This heuristic is required because the machine separation constraint becomes difficult to satisfy by the penalty method. The Repair Heuristic first arbitrarily forms a certain number of mutually exclusive sets of machines taken from S such that the machines in a given set can be placed in the same cell. Once these sets of machines are identified, a certain number of cells will be associated with each set. A cell will be associated with at most one set of machines while a set can be associated with more than one cell. Finally, for a chromosome under repair, operations that require a machine in a given set are arbitrarily assigned to one of the cells associated with that particular set. This guarantees the fulfillment of machine separation constraint. The heuristic has randomness behavior making it compatible with the genetic search. The heuristic has been explained here under in an algorithmic format using the following notations.

Θ'	Set of machines (unpaired) included in Θ,
ϑ, φ	Counter variables,
$\Theta_{\vartheta\varphi}$	Set of machines from which φ^{th} machine of the ϑ^{th} set is to be arbitrarily taken,
$k_{\vartheta\varphi}$	The φ^{th} machine of the ϑ^{th} set arbitrarily taken from $\Theta_{\vartheta\varphi}$,
Δ	A function that take a set of machines $\Theta_{\vartheta\varphi} \setminus \{k_{\vartheta\varphi}\}$ and machine $k_{\vartheta q}$ as the first and second arguments respectively and returns a set of machine $\Theta_{\vartheta,\ \varphi+1}$, taken from its first argument, that can be placed in the same cell with machine type $k_{\vartheta\varphi}$,
Υ_{ϑ}	The ϑ^{th} set of machines taken from Θ' that can be placed in the same cell

Using the above notations, the algorithm for Repair-Heuristic is presented below. A simple example will follow to make the heuristic more clear.

Step 1. Set $\vartheta = 1$ and $\varphi = 1$ and $\Theta_{1,1} = \Theta'$.

Step 2. Arbitrarily take machine k_{pq} from Θ_{pq}.

Step 3. Determined $\Theta_{\vartheta,\ \varphi+1} = \Delta(\Theta_{pq} \setminus \{k_{pq}\},\ k_{pq})$

Step 4. If $\Theta_{\vartheta,\ \varphi+1} \neq \emptyset$, set $\varphi = \varphi + 1$ and go to step 2; otherwise go to step 5.

Step 5. $\Upsilon_{\vartheta} = \{k_{\vartheta,1},\ k_{\vartheta,2},\ \ldots k_{pq}\}$ forms the ϑ^{th} set of machines arbitrarily taken from Θ' that can be placed in the same cell. Set $\Theta_{\vartheta+1,1} = \Theta_{\vartheta,1}/\Upsilon_{\vartheta}$. If $\Theta_{\vartheta+1,1} \neq \emptyset$, set $\vartheta = \vartheta + 1$ and $\varphi = 1$ and go to step 3b; otherwise go to step 3f.

Step 6. At this step ϑ equal to the number of sets of machines taken from Θ which can be placed in the same cell. Randomly and evenly associate the cells with the ϑ sets identified such that a cell is associated with at most one set and each set is associated with minimum of L mod ϑ cells.

Step 7. For a chromosome under repair, assign operations that require a machine in a given set to one of the cells associated with that particular set.

The following example demonstrates the application of Repair-Heuristic on a chromosome violating machine separation constraint. For this example let us assume that the set of machine pairs that cannot be located in the same cell be given by Θ = {(1, 2), (1, 4), (2, 5), (4, 7), (2, 8)}. Let us also assume that the first operation of part 1 requires machine 1, the third operation of part 3 requires machine 4 and the fifth operation of part 4 requires machine 7. Consider a chromosome that assigns

all of these operations to cell 1. Then it is obvious that this chromosome violates machine separation constraint since it requires the placement of machines 1, 4 and 7 in cell 1. Hence, Repair-Heuristic has to be applied on this chromosome. A step by step execution of this heuristic is shown below.

Θ = {(1, 2), (1, 4), (2,5), (4, 7), (2, 8)}

Θ' = {1, 2, 4, 5, 7, 8}

Step 1. $\vartheta = 1, \quad \varphi = 1, \quad \Theta_{1,1} = \Theta' = \{1, 2, 4, 5, 7, 8\}$.

Step 2. $k_{1,1} = 4$ (arbitrarily selected from $\Theta_{1,1}$).

Step 3.
$$\begin{aligned} \Theta_{1,2} &= \Delta(\Theta_{1,1}/\{k_{1,1}\},\ k_{1,1}) \\ &= \Delta(\{1, 2, 5, 7, 8\},\ 4) \\ &= \{2,\ 5,\ 8\}. \end{aligned}$$

Step 4. $\Theta_{1,2} \neq \emptyset$. Hence, set $\varphi = \varphi + 1 = 1 + 1 = 2$ and go to step 2.

Step 2. $k_{1,2} = 5$ (arbitrarily selected from $\Theta_{1,2}$).

Step 3.
$$\begin{aligned} \Theta_{1,3} &= \Delta(\Theta_{1,2}/\{k_{1,2}\},\ k_{1,2}) \\ &= \Delta(\{2, 8\},\ 5) \\ &= \{8\}. \end{aligned}$$

Step 4. $\Theta_{1,3} \neq \emptyset$. Hence, set $\varphi = \varphi + 1 = 2 + 1 = 3$ and go to step 2.

Step 2. $k_{1,3} = 8$ (arbitrarily selected from $\Theta_{1,3}$).

Step 3.
$$\begin{aligned} \Theta_{1,4} &= \Delta(\Theta_{1,3}/\{k_{1,3}\},\ k_{1,3}) \\ &= \Delta(\emptyset,\ 8) \\ &= \emptyset. \end{aligned}$$

Step 4. $\Theta_{1,4} = \emptyset$. Hence, go to step 5.

Step 5. $\Upsilon_1 = \{k_{1,1},\ k_{1,2},\ k_{1,3}\}$

$= \{4,\ 5,\ 8\}$

$\Theta_{2,1} = \Theta_{1,1}/\Upsilon_1$

$= \{1,\ 2,\ 7\}$

$\neq \emptyset$. Hence, set $\vartheta = \vartheta + 1 = 1 + 1 = 2$, $\varphi = 1$ and go to step 2.

Step 2. $k_{2,1} = 1$ (arbitrarily selected from $\Theta_{2,1}$).

Step 3. $\Theta_{2,2} = \Delta(\Theta_{2,1}/\{k_{2,1}\},\ k_{2,1})$

$= \Delta(\{2,7\},\ 1)$

$= \{7\}$.

Step 4. $\Theta_{2,2} \neq \emptyset$. Hence, set $\varphi = \varphi + 1 = 1 + 1 = 2$ and go to step 2.

Step 2. $k_{2,2} = 7$ (arbitrarily selected from $\Theta_{1,3}$).

Step 3. $\Theta_{2,3} = \Delta(\Theta_{2,2}/\{k_{2,2}\},\ k_{2,2})$

$= \Delta(\emptyset,\ 7)$

$= \emptyset$.

Step 4. $\Theta_{2,3} = \emptyset$. Hence, go to step 5.

Step 5. $\Upsilon_2 = \{k_{2,1},\ k_{2,2}\}$

$= \{1,\ 7\}$

$\Theta_{3,1} = \Theta_{2,1}/\Upsilon_2$

$= \{2\}$

$\neq \emptyset$. Hence, set $\vartheta = \vartheta + 1 = 2 + 1 = 3$, $\varphi = 1$ and go to step 2.

Step 2. $k_{3,1} = 2$ (arbitrarily selected from $\Theta_{3,1}$).

Step 3.
$$\begin{aligned}\Theta_{3,2} &= \Delta(\Theta_{3,1}/\{k_{3,1}\},\ k_{3,1}) \\ &= \Delta(\emptyset,\ 2) \\ &= \emptyset.\end{aligned}$$

Step 4. $\Theta_{3,2} = \emptyset$. Hence, go to step 5.

Step 5.
$$\begin{aligned}\Upsilon_3 &= \{k_{3,1}\} \\ &= \{2\} \\ \Theta_{4,1} &= \Theta_{3,1}/\Upsilon_3 \\ &= \emptyset.\ \text{Hence, go to step 6.}\end{aligned}$$

Step 6. In the previous steps $\vartheta = 3$ groups are identified. Assuming in the problem at hand four cells ($L = 4$) are to be generated, allocated the four cells randomly and evenly to the three groups identified. Let cell 3 be allocated to group 1, cells 1 and 2 to group 2 and cell 4 to group 3.

Step 7. For the example chromosome under consideration assign operations requiring either of machines 4, 5, or 8 to cell 3, each of those requiring either of machines 1 or 7 to cell 1 or 2, and those requiring machine 2 to cell 4. By doing so the chromosome now satisfies machine separation constraint.

4.2.6. Machine Assignment Heuristic

The continuous variables $\eta_{jikl}(t)$ and $\bar{\eta}_i(t)$ are determined by decoding a chromosome. The corresponding integer variables $N_{kl}(t)$, $y^+_{kl}(t)$ and $y^-_{kl}(t)$ are determined using the Machine Assignment Heuristic. The heuristic was applied for each individual chromosome so that the corresponding value of the fitness function can be

evaluated. Once the values of the continuous variables $\eta_{jikl}(t)$ and $\bar{\eta}_i(t)$ are known, the minimum number of each type of machines $M_{kl}(t)$ to satisfy capacity requirement in each cell during each period is determined by the following equation:

$$M_{kl}(t) = \left\lceil \frac{\sum_{i=1}^{P} \sum_{j=1}^{J_i} (d_i(t) \cdot \eta_{jikl}(t) \cdot h_{jik})}{C_k} \right\rceil \tag{4.5}$$

where $\lceil x \rceil$ returns the smallest integer greater or equal to x.

After this step, the heuristic sets the number of machines, $N_{kl}(1)$, in each cell for period 1 equal to $M_{kl}(1)$. The number of machines of each type installed in the various cells for $t > 1$ are determined as follows. Let $M_{kl}(t)$ and $\widetilde{M}_k(t)$ represent the minimum number of type k machines required in cell l and in the system respectively and $\widetilde{N}_k(t)$ represents the actual number of type k machines installed in the system during period t. If $\widetilde{N}_k(t-1) \leq \widetilde{M}_k(t)$, then the heuristic sets $N_{kl}(t) = M_{kl}(t)$. If $\widetilde{N}_k(t-1) > \widetilde{M}_k(t)$, then heuristic assigns those type k machines to the various cells to satisfy the required minimum number of machines in period t and leaves the extra machines in the cells in which they were perviously installed. This process of determining the number of machines of each type installed in each cell for $t > 1$ can be better illustrated using the pseudocode given in Fig. 4.3.

To further illustrate, we present a simple example to place type 5 machines in a 4-cell system for 4 time periods. The minimum number of type 5 machines required in each cell for each period are shown in Table 4.1. For period 1, we have $N_{5,l}(1) = M_{5,l}(1)$. Hence, $N_{5,1}(1) = 2$, $N_{5,2}(1) = 1$, $N_{5,3}(1) = 3$, $N_{5,1}(1) = 2$ and $N_{5,4}(1) = 0$. The following step by step application of the pseudocode gives $N_{5,l}(t)$ for $t > 1$.

FOR^1 : $k = 5$

FOR^2 : $t = 1$

```
Declare new integer variable called Buffer and Add
FOR[1]  k = 1 to K
    FOR[2]  t = 1 to T − 1
        N̂_k(t) = 0
        M̂_k(t+1) = 0
        FOR[3]  l = 1 to L
            N̂_k(t) = N̂_k(t) + N_kl(t)
            M̂_k(t+1) = M̂_k(t+1) + M_kl(t+1)
        END FOR[3]
        IF[1]  M̂_k(t+1) < N̂_k(t)
            Buffer = N̂_k(t) − M̂_k(t+1)
            FOR[4]  l = 1 to L
                IF[2]  M_kl(t+1) < N_kl(t)
                    Add = min{Buffer,  N_kl(t) − M_kl(t+1)}
                    N_kl(t+1) = M_kl(t+1) + Add
                    Buffer = Buffer − Add
                END IF[2]
                ELSE[2]
                    N_kl(t+1) = M_kl(t+1)
                END ELSE[2]
            END FOR[4]
        END IF[1]
        ELSE[1]
            FOR[5]  l = 1 to L
                N_kl(t+1) = M_kl(t+1)
            END FOR[5]  l = 1 to L
        END ELSE[1]
    END FOR[2]
END FOR[1]
```

Figure 4.3: Psudocode of step 4 of the machine assignment heuristic

Initialize $\widetilde{N}_5(1) = 0$ and $\widetilde{M}_5(2) = 0$

FOR[3] : $l = 1$ to 4 (performing successive addition)

$\widetilde{N}_5(1) = 7$ and $\widetilde{M}_5(2) = 4$

IF[1] : The logical test $\widetilde{M}_5(2) = 4 < \widetilde{N}_5(1) = 7$ is TRUE

$Buffer = \widetilde{N}_5(1) - \widetilde{M}_5(2) = 7 - 4 = 3$

FOR[4] : $l = 1$

IF[2] : The logical test $M_{5,1}(2) = 0 < N_{5,1}(1) = 2$ is TRUE

$Add = min\{Buffer, N_{5,1}(1) - M_{5,1}(2))\} = min\{3, 2 - 0\} = 2.$

Table 4.1
Minimum number of type 5 machine to meet capacity requirement

l	$M_{5,l}(1)$	$M_{5,l}(2)$	$M_{5,l}(3)$	$M_{5,l}(4)$
1	2	0	1	1
2	1	2	4	3
3	3	1	2	0
4	0	1	2	3
$\widetilde{M}_5(t)$	7	4	9	7

$N_{5,1}(2) = M_{5,1}(2) + Add = 0 + 2 = 2$

$Buffer = Buffer - Add = 3 - 2 = 1$

FOR^4 : $l = 2$

IF^2 : The logical test $M_{5,2}(2) = 2 < N_{5,2}(1) = 1$ is FALSE

ELSE^2 : $N_{5,2}(2) = M_{5,2}(2) = 2$

FOR^4 : $l = 3$

IF^2 : The logical test $M_{5,2}(2) = 1 < N_{5,1}(1) = 3$ is TRUE

$Add = min\{Buffer, N_{5,2}(1) - M_{5,2}(2))\} = min\{1, 3 - 1\} = 1.$

$N_{5,3}(2) = M_{5,3}(2) + Add = 1 + 1 = 2$

$Buffer = Buffer - Add = 1 - 1 = 0$

FOR^4 : $l = 4$

IF^2 : The logical test $M_{5,4}(2) = 1 < N_{5,4}(1) = 0$ is FALSE

ELSE^2 : $N_{5,4}(2) = M_{5,4}(2) = 1$

From the above steps

$N_{5,1}(2) = 2$, $N_{5,2}(2) = 2$, $N_{5,3}(2) = 2$ and $N_{5,4}(2) = 1$

FOR^2 : $t = 2$

Initialize $\widetilde{N}_5(1) = 0$ and $\widetilde{M}_5(2) = 0$

FOR^3 : $l = 1$ to 4 (performing successive addition)

$\widetilde{N}_5(2) = 7$ and $\widetilde{M}_5(3) = 9$

IF[1] : The logical test $\widetilde{M}_5(3) = 9 < \widetilde{N}_5(2) = 7$ is FALSE

ELSE[1] :

FOR[5] : $l = 1$ to 4 (performing successive assignment operation)

$N_{5,1}(3) = M_{5,1}(1) = 1$, $N_{5,2}(3) = M_{5,1}(2) = 4$

$N_{5,3}(3) = M_{5,1}(3) = 2$, $N_{5,4}(3) = M_{5,1}(4) = 2$

Continuing the above computation for $t = 3$, we can determine $N_{5,l}(t)$ for $t = 4$. The results of the above computations are summarized in Table 4.2. From Table 4.2, one can see that $N_{5,l}(t)$ is greater than or equal to the required number of machines $M_{5,l}(t)$ shown in Table 4.1. Hence, the machine capacity constraint, Eq. (3.6), is satisfied. At the same time, we have $\widetilde{N}_5(t') \geq \widetilde{N}_5(t)$ for $t' > t$. This guarantees the fulfilment of the constraint in Eq. (3.8) of the model. Once the decision variables $N_{kl}(t)$ are determined, the configuration decision variables $y_{kl}^{+}(t)$ and $y_{kl}^{-}(t)$ can be determined using Eqs. (4.6) and (4.7) respectively. This last step satisfies the constraint in Eq. (3.10) of the proposed model.

$$y_{kl}^{+}(t) = \begin{cases} N_{kl}(t), & \text{if } t = 1, \\ max\{0,\ N_{kl}(t) - N_{kl}(t-1)\}, & \text{if } t > 1. \end{cases} \tag{4.6}$$

$$y_{kl}^{-}(t) = \begin{cases} 0, & \text{if } t = 1, \\ max\{0,\ N_{kl}(t-1) - N_{kl}(t)\}, & \text{if } t > 1. \end{cases} \tag{4.7}$$

Table 4.2
Actual number of type 5 machines installed

l	$N_{5,l}(1)$	$N_{5,l}(2)$	$N_{5,l}(3)$	$N_{5,l}(4)$
1	2	2	1	1
2	1	2	4	4
3	3	2	2	1
4	0	1	2	3
$\widetilde{N}_5(t)$	7	7	9	9

4.2.7. Constraints Handling

[102] stated that the central problem in the application of genetic algorithms is constraint handling. He proposed different approaches to handle constraints such as rejection of infeasible solutions (death penalty), penalty methods and repair algorithms. In this study, the latter two and other approaches were used to handle constraint violations. Constraints in Eqs. (3.7), (3.9) and (3.14) are handled using the penalty method. This method adds penalty quantities if corresponding constraints are not satisfied. A problem specific Repair Heuristic is developed to handle the constraints in Eqs. (3.11)–(3.13). Constraints in Eqs. (3.2)–(3.5) are handled by the use of the chromosomal structure to generate solutions satisfying these constraints. The Machine Assignment Heuristic handles constraints in Eqs. (3.6), (3.8) and (3.10). Constraints in Eqs. (3.15) is handled by genetic operators which assign values in [0, 1] to $\bar{\eta}_i(t)$ and x_j's. The integrality constraint given by Eq. (3.16) is handled by the Machine Assignment Heuristic and the Repair Heuristic.

4.2.8. The Two-Phase Genetic Algorithm

Generating independent cells is computationally simpler than generating cells which optimize inter-cell movements and other cost terms of the objective function.

In the solution space, the best configuration with independent cells is a neighborhood solution to the best configuration with optimal inter-cell movements. In the first phase, the algorithm attempts to quickly find the best configuration with independent cells. To perform this task, the algorithm applies genetic operators which do not produce solutions having inter-cell movement. These include all the operators other than the single-point crossover and the operation-level cell mutator. Inter-cell movement can occur during the first phase only as a result of the Repair Heuristic. In the second phase, the algorithm finds the configuration with best inter-cell movements. All the genetic operators other than the part-level cell mutator are applied during the second phase. In addition to running in two phases, the proposed genetic algorithm has a procedure called population rejuvenation. This procedure reduces and focusses the search domain around the best solution found if this value does not improve in a given number of successive generations. After a given number of successive generations without any improvement of the incumbent solution, we consider that the genetic algorithm has detected the promising area. A certain number of individuals of the current population will be replaced by the duplicates of the best individual so far found. Hence, the population diversity will be reduced and centered around the best solution. This process rejuvenates the population which may have been deteriorated by a large number of applications of the genetic operators. The following symbols and notations used are present the steps of the genetic algorithm.

g	Generation counter, $g = 1, 2, ..., g_{max}$,
g_{max}	Maximum number of generations,
$PP(g)$	Parents population of generation g,
$CP(g)$	The current children population which makes up $PP(g+1)$,
BI	Best feasible individual so far found,

$Phase$ The current phase of the search which equals to 1 for the first phase or 2 for the second phase,

g_{phase} Generation at which the value of $Phase$ should be set equal to 2 if it were not previously set to this value by other conditions,

w Number of successive generations counted without any improvement of the best individual so far found,

w_{max} Maximum value of w at which point population rejuvenation is to be performed,

b Number of successive population rejuvenations counted without any improvement of the best individual so far found,

b_{max1} Maximum value of b at which point the second phase is to be entered if $Phase$ was equal to 1,

b_{max2} Maximum value of b in the second phase at which point the search will be terminated.

Using the above notations, the steps of the proposed genetic algorithm based heuristic are listed below. These steps were coded using C++ programming language and run on a Pentium-4 (2.4 GH, 768 MB Ram) PC computer.

Step 1. Initialize the counters g, w, b and $Phase$ such that $g = w = b = 0$ and $Phase = 1$. Initialize the Best-Individual BI so far found with a null value.

Step 2. Randomly generate initial parent-population $PP(0)$ such that each individual represents a solution having independent cells (i.e. the c_j's of all the operations of a given part are set equal),

Step 3. Apply the Repair Heuristic (Steps 3a to 3g) for each individual chromosome violating machine separation constraint.

Step 4. For each chromosome, using the Machine Assignment Heuristic, determine the number of machines of each type assigned to the various cells during each period.

Step 5. For each individual, determine $\tilde{F}$. If an improvement has been obtained update the previous BI by the current BI, set $w = b = 0$; otherwise increase w by one.

Step 6. If $w = w_{max}$, rejuvenate the population by replacing k% of the population by the duplicates of previous BI and reset $w = 0$ and increase b by one.

Step 7. If $b = b_{max1}$ and $Phase = 1$ set $Phase = 2$ and $w = b = 0$.

Step 8. Select individuals from the current parent-population to become parents of the next generation according to their fitness value.

Step 9. Randomly mate parent-population and create children-population $CP(g)$ by applying genetic operators valid to the current phase of the search.

Step 10. Replace the current parent-population $PP(g)$ by the children-population $CP(g)$ and form the new parent-population $PP(g+1)$. Increase the generation counter, $g = g + 1$. If $g = g_{phase}$ and $Phase = 1$, then set $Phase = 2$.

Step 11. If the termination criterion has not been met, go to step 3. Termination criterion is $g = g_{max}$, or $b = b_{max2}$ for $phase = 2$.

4.3. Parallelizing the GA

Sequential GAs (SGAs) have been shown to be very successful in many applications and in very different domains. However there exist some problems in their

utilization. In certain problems, fitness evaluation can be a very time-consuming process (e.g. if it is determined by using simulation, by solving certain sub-problems, e.t.c). For some other problems, the population needs to be very large and the memory required to store each individual may be considerable. SGAs might be trapped in a sub-optimal region of the search space as they use and evolve only a single population. These problems of the SGAs can all be addressed with some form of Parallel GA (PGA). The parallelism of fitness evaluations was the first approach for PGA, called the Master-Slave PGA (MSPGA). This model uses a single global population and the fitness evaluation is done on different processors. Furthermore, crossover and mutation operations may also be done in parallel. The nature of GA is not changed because the selection operation is done globally with the whole population using the master computer. This parallelism can be very useful when the fitness evaluation is a time consuming process. The other types of parallelism, called distributed PGAs (DPGAs), use multiple populations maintained by different processors. These can be classified as coarse-grained and fine-grained DPGA. The coarse-grained DPGAs use few large sub-populations and can be further classified in to two categories [110]: those using sub-populations with migration of individuals from one subpopulation to another (*island model*) and those with overlapping sub-populations where genetic materials are exchanged via the overlapping portions of the sub-populations. The island model can easily be implemented on a multiple-memory-multiple-processor architecture or network of workstations while the overlapping subpopulation approach is suitable for a share memory architectures. Fine-grained DPGAs use a population separated into a large number of very small sub-populations, which are maintained by different processors. The subpopulation may be only an individual. This model is suitable for massively parallel architectures - machines consisting of a huge number of basic processors and connected with a specific high speed topology.

The psudocodes for SGA and DPGA are given in Fig. 4.4. The two psudocodes are essentially the same except for the communication routines of DPGA. Hence, only a little effort is needed for converting sequential GA to coarse-grained PGA. The complete algorithm of DPGA consists of multiple copies of its sub-algorithm being excused in parallel.

```
Initialize population
Repeat g = 1, 2, 3, ....
   Evaluate solutions in the population
   Perform competitive selection
   Apply genetic operators
Until convergence criterion satisfied
```

(a) SGA

```
Initialize subpopulation
Repeat g = 1, 2, 3, ....
   Evaluate solutions in the population
   Perform competitive selection
   If it is time to communicate {
      Select migrants
      Send migrants to destinations
      Receive migrants from sources
   }
   Apply genetic operators
Until convergence criterion satisfied
```

(b) DPGA Sub-algorithm

Figure 4.4: Psudocode for sequential and distributed parallel genetic algorithms.

DPGAs have an appealing trait in that they often reduce the computational effort to solve the same problem as compared to SGAs, even when run in a single processor [46]. This characteristic makes a difference with respect to other searching algorithms, in that DPGAs are not simple parallel versions of sequential algorithms. Thus they represent a new class of algorithms in that they search the solution space

differently ([110]). The reason for this can be found in the most striking characteristics of DPGAs [3] as (1) their decentralized search, which allows speciation (different sub-populations evolve towards different solutions), (2) the larger diversity levels (many search regions are sought at the same time) and (3) exploitation inside these sub-populations, i.e., refining the better partial solutions found in each sub-populations.

4.3.1. The Proposed Coarse-Grained DPGA

Being motivated by the characteristics of DPGA mentioned in the previous section, we develop a coarse-grained DPGA for designing cellular manufacturing systems based on the island model. The island model is selected because this model is easily implemented on a clustered computers using public domain libraries such as MPI and PVM. In this model, the migration of individuals from one subpopulation to another is controlled by two parameters. The first is the topology that defines the connection among the sub-populations. Commonly used topologies are ring, two dimensional mesh, hypercubes and fully connected. In this study we use fully connected topologies shown in Figs 4.5.

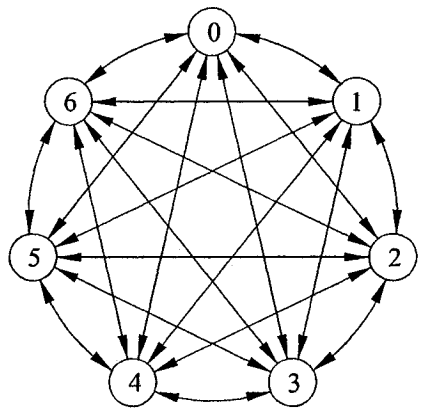


Figure 4.5: Fully connected topology

The second parameter, controlling the migration of individuals, is the migration operator. This operator is composed of a number of attributes like: (1) the number of individuals undergoing migration, (2) the frequency of migration in numbers of generations and (3) the migration scheme, that controls which individuals from the source subpopulation (best, according to fitness, random, etc.) migrate to another subpopulation, and which individuals are replaced (worst, random, etc.). The incoming individuals are combined with a subpopulation after the selection and before the crossover and mutation operators are applied in the destination subpopulation.

4.4. Computational Performance of Proposed GA

4.4.1. Computational Performance of SGA

The performance of the developed heuristic was evaluated against the results from LINGO. As can be seen from Table 4.3, the lower bound and the best objective function value found by LINGO after 5 hours of computation were 1.69346 and

1.76865 respectively. From this table and Fig. 4.6, it can be seen that the developed heuristic was able to determine a solution having an objective function value equal to 96% of the lower bound in less than 20 minutes. The figure also shows that the average of the objective function values of the feasible solutions varies from generation to generation and is far from the best objective function value found. This indicates that the genetic algorithm has maintained diversity of the population in performing a global search. In addition to maintaining the population diversity, the genetic algorithm has converged to an acceptable solution very quickly. This was achieved by using those problem specific procedures as systematically dividing search phases, population rejuvenation, constraint handling and other related genetic operators. As shown in Figs. 4.7 and 4.8, dividing the search into phases, populations rejuvenating and using the degenerator operators enable the genetic algorithm to find near optimal solutions very quickly. The values of the various parameters of the GA used for this computation are given in Table B.1 under Test 1. The experiment was repeated under other 12 parameter settings given in table B.1 under the Tests 2 though 13. The convergence graphs of the GA for these tests are given figure 4.9. From this figure it can be seen that the GA was able to converge to an acceptable solution for several parameter settings. We were also able to find several other parameter settings for which the genetic algorithm performs very well. This can be an indication for the relative robustness of the proposed method.

Table 4.3
Comparison of the proposed GA with LINGO

Time	Lower Bound	Objective value using	
hr:min:sec	determined by LINGO	LINGO	the proposed GA
00:00:15	-	×	1.933360
00:00:30	1.69274	×	1.785400
00:01:00	1.69274	×	1.781189
00:02:00	1.69274	×	1.779745
00:06:00	1.69274	1.86962	1.776739
00:08:00	1.69275	1.86962	1.762127
00:10:00	1.69277	1.86962	1.761973
00:12:00	1.69282	1.81068	1.761863
00:16:00	1.69282	1.81068	1.761793
00:20:00	1.69309	1.81068	1.761793
00:30:00	1.69311	1.77401	1.761793
01:00:00	1.69323	1.76934	*
03:00:00	1.69344	1.76934	*
05:00:00	1.69346	1.76865	*
10:00:00	1.69353	1.76865	*
15:00:00	1.69366	1.76865	*
20:30:00	1.69377	1.76865	*

× - Feasible solution was not found
* - A termination criterion was met.
Note: Values are in Millions.

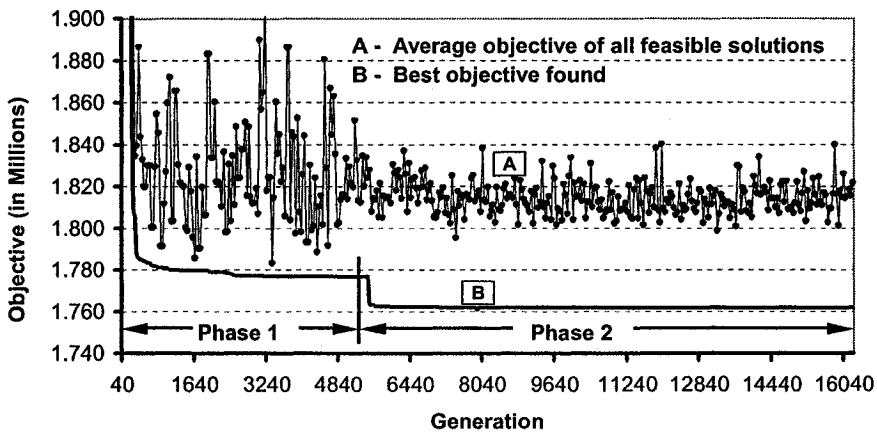


Figure 4.6: A 15-minutes convergence history of the proposed GA.

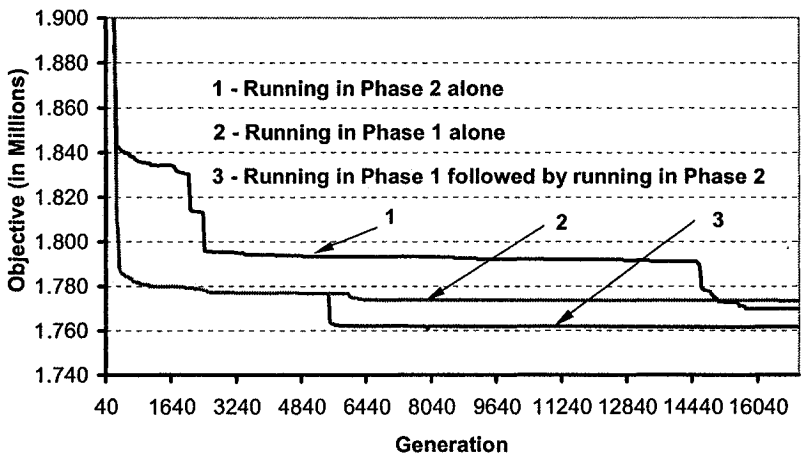


Figure 4.7: The effect of dividing the search into phases

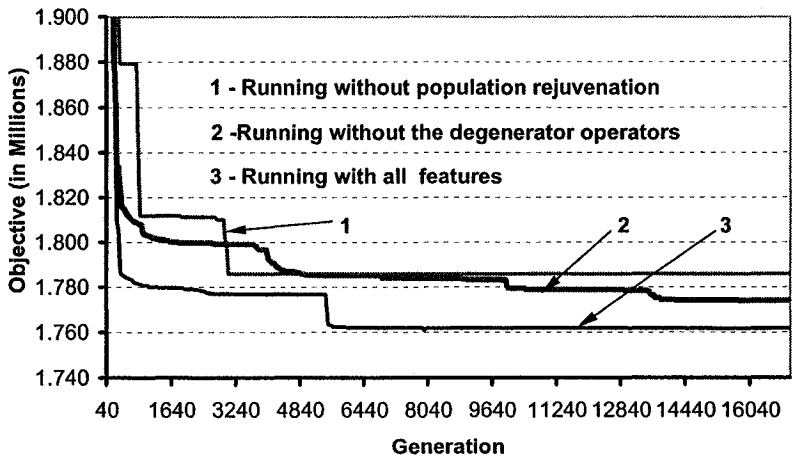


Figure 4.8: The effects of population rejuvenation and degenerator operators

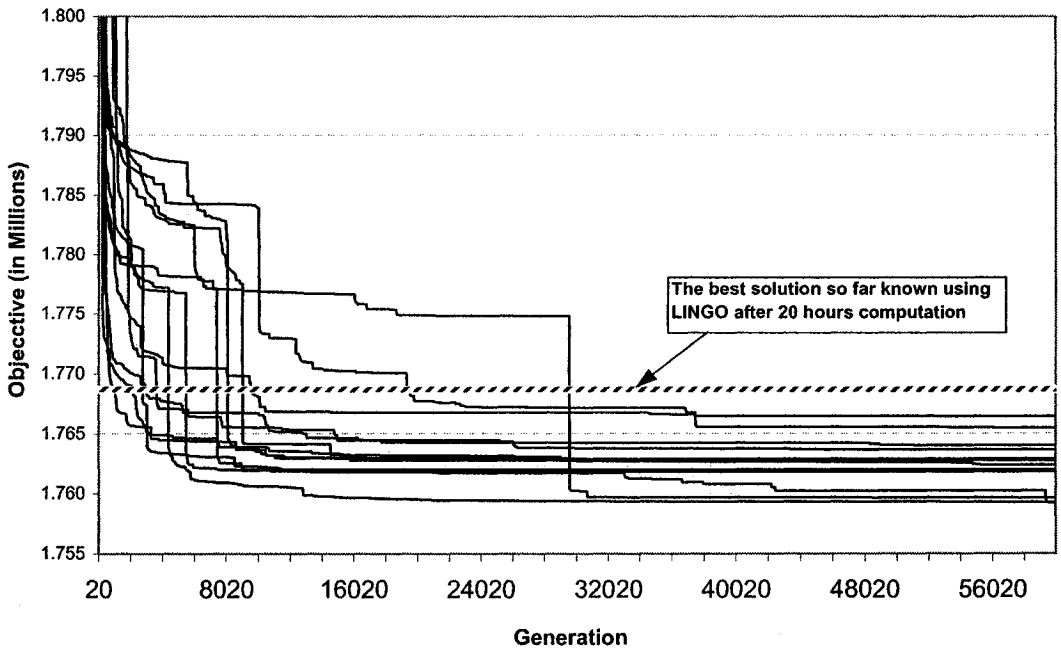


Figure 4.9: Less than an our convergence history of the proposed GA using 13 different parameter settings given in table B.1

4.4.2. Performance Improvement through Parallelization

The proposed parallel GA was coded in C++ programming language using MPI message-passing library for communication. The code was tested in a parallel computation environment composed of 872 waiters containing each one Pentium 4 processor (3.2 GHz, 2GB RAM). The test problem was run using only 1, 2, 4, 8, 12, and 16 processors with a subpopulation size of 460. Each sub-algorithm uses separately seeded pseudo-random number generator to promote the sub-algorithms exploring different parts of the search space. Migration was taking place every 40 generation (migration frequency) and the number of migrants M (the migration rate) was determined in such a way that the total number of migrants joining a given subpopulation is about equal to 10% of its size. To meet this requirement in a fully connected topology, where each subpopulation receives M migrants from each other subpopulation, M was determined such that $(P-1) \times M \approx 0.1 \times 460$ where P is the total number of sub-populations (or processors). Migrants were selected randomly from the source subpopulation and the individuals in a destination subpopulation were replaced randomly by the incoming migrants. The total number of individuals evaluated per generation is equal to $P \times 460$. Our observations as P increases was that, by mapping each subpopulation to each processor, the total elapsed time remain relatively constant (except for the parallel computing overheads associated with the migration of individuals, which is relatively very small in the proposed coarse-grained PGA). A run was terminated after 32,000 generation. Fig. 4.10 shows the convergence graphs of SGA and DPGA with different number of processors. The graph shows that increasing the number of processing nodes help much on improving convergence. It also shows that the difference between the results with parallel and serial algorithm is more evident. This can be explained by the various features of

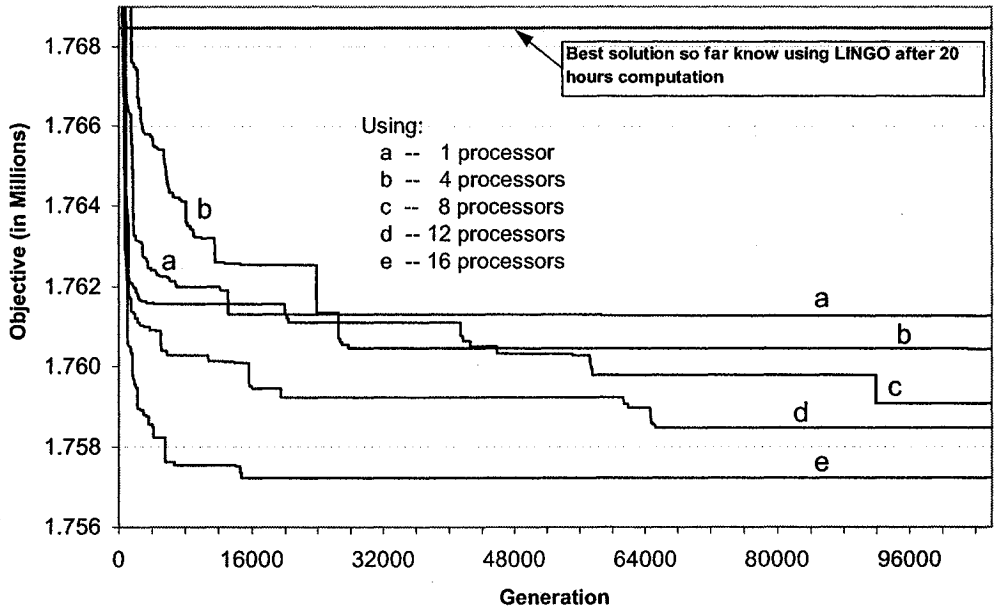


Figure 4.10: Convergence graphs using different number processors

DPGA mentioned in Section 4.3.

The performance of the proposed island model was also evaluated on single machine having only one processor (2.4 GH, 720 MB Ram) using MPICH[1]. The implementation of the island PGA on single processor machine consists of autonomous processes[2], each executing its own code in its own address space and evolve its own subpopulation. The processes communicate via calls to MPI communication primitives, similar to the actual parallel implementations. The convergence graph was similar to the one shown in Fig. 4.10, except the DPGA requires longer exclusion time. As an example, the times required to move 32,000 generations by the SGA and the eight-subpopulation DPGA are 00:24:30 and 04:07:36 respectively on a single processor machine. Though the DPGA requires longer time to run the whole 32,000 generations on such single processor machine, it was able to give better solutions in

[1]A freely available, portable implementation of MPI
[2]Identical codes running simultaneously.

few generations than those can be found using SGA.

4.5. Summary and Discussions

In the previous chapter, we present a comprehensive mathematical programming model for the formation of manufacturing cells over multiple time periods. For small size problems, off-the-shelf optimization software LINGO was used to solve this formulation. For solving large size problems, branch and bound based general search algorithm employed by such software cannot give optimal or near optimal solutions within acceptable computational times on widely available platforms such as a PC computer. To this end, in this chapter, we developed an efficient heuristic based on genetic algorithm. The proposed heuristic runs in two phases. In the first phase, independent cells are formed which are relatively simple to generate. In the second phase, the solution found during the first phase is gradually improved to generate cells optimizing inter cell movement and other cost terms of the model. The algorithm also incorporates several problem specific genetic operators and constraint handling techniques. The proposed heuristic was evaluated by comparing its computational results and those from available optimization software. The results obtained by using the heuristic method were very encouraging. In order to further improve the obtained results, a parallel implementation of the algorithm (PGA) is also attempted.

In literature, there have been a few report on the use of PGA for CMS design. To our knowledge the use of PGA in CMS design was reported only in [12]. As yet, the PGA methodology has not been exploited for the design of CMSs and so

this novel approach is attempted in this chapter. The parallel algorithm was implemented based on the island model on parallel computing environment composed of a cluster of workstations. The parallel implementation of the algorithm demonstrates a reduction of processing time and improved search performance, even when it is implemented on a single machine with one processor. Thus, with this work, we could ratify the importance of using parallel genetic algorithm in CMS design where there are a few reports on its use. The coarse-grained island model PGA is selected with three criteria: (1) It is suitable for a clustered computer, which is easily implemented from cheap PCs connected through a low cost network and using public domain software; (2) A little effort is needed for converting sequential GA to coarse-grained PGA since each processor performs simple GA except periodically exchanges some population by migration operators; (3) It can be implemented on a single processor machine and perform better than SGA. We evaluate the performances of SGA and coarse-grained PGA with varying a number of processing nodes. The convergence graphs were used to compare the parallel algorithm with the serial one. The difference between the empirical efficiency of the serial and the parallel algorithm is clear. The later demonstrates an improved search performance.

Chapter 5

SA Based Solution Procedure for Math Model-A

5.1. Introduction

Simulated annealing is a general-purpose optimization technique capable of finding optimal or near-optimal solutions in various applications. The major disadvantage of this technique is its slow convergence making it not suitable for solving many complex optimization problems. This limitation may be alleviated by parallel computing using a multiprocessor computer or a cluster of workstations. In this chapter, we developed and presented a multiple Markov chain simulated annealing algorithm which allows multiple search directions to be traced simultaneously. Our computational results on a single processor machine showed that multiple Markov chain SA is much more efficient than a conventional single Markov chain SA. The parallel implementation of the multiple Markov chain SA greatly improved its computational efficiency in terms of solution quality and execution time.

5.2. Simulated Annealing Based Heuristics

5.2.1. The Basic Algorithm

Kirkpatrick *et al.* [79] first introduced the general optimization algorithm of simulated annealing based on the work of Metropolis *et al.* [101]. It is a stochastic algorithm generating a sequence X_0, X_1, $\cdots$, X_n, $\cdots$ of random solutions of a problem approaching the set of optimal solutions as $n \rightarrow \infty$. This sequence of solutions is generated by following Eq. 5.1 where X'_n is a neighborhood solution generated by slightly perturbing X_n and $E(X_n)$ is the value objective function.

$$X_{n+1} = \begin{cases} X'_n & \text{if } E(X'_n) \leq E(X_n) \\ X'_n & \text{if } \exp\left(\frac{E(X_n)-E(X'_n)}{T_n}\right) > \text{rand}() \\ X_n & \text{otherwise} \end{cases} \tag{5.1}$$

In the above equation, rand() is a random number generator for making a stochastic decision for the new solution. $T_n > 0$ is the *temperatures* at the n^{th} and the sequence of T_n such that $T_n \geq T_{n+1}$ with the $\lim_{n\rightarrow\infty} T_n = 0$ is the *cooling schedule.* From Eq. 5.1, it can be seen that the choice of X_{n+1} depends only on the current solution X_n but not on the previously visited solutions. Thus, the search path of such simulated annealing algorithms follows a Markov chain. From here onwards we refer the basic algorithm to as a Sequential Single Markov Chain Simulated Annealing Algorithm (SSMC-SA). For a suitable cooling schedule, the probability of the error of not getting an optimal solution after n iterations using SSMC-SA is characterized by Eq. 5.2 where U_{min} is a set of optimal solution points [34]. In this equation, n is large enough and $K > 0$ and $\theta > 0$ are constants peculiar to a given energy

landscape and neighborhood generation. Thus as $n \rightarrow \infty$, the solution converges to one of the optimal points in U_{min} with a probability of 1.

$$P(X_n \notin U_{min}) \sim \left(\frac{K}{n}\right)^{\theta} \tag{5.2}$$

5.2.2. Sequential Multiple Markov Chain SA

Most SA schemes in the literature follow a single Markov chain. However, following a single Markov chain may not be necessary from performance point of view [86]. A Sequential Multiple Markov Chain SA (SMMC-SA) performs S independent versions of annealing on a single processor computer, using the same search space, neighborhood generation and cooling schedule. Each one of this independent version is stopped after n iterations to provide S independent terminal solutions $\{X_{n,1}, X_{n,2}, \cdots, X_{n,S}\}$. Then out of these terminal solutions, the best one is chosen as the final solution X_n. A psudocode for an instance of such Sequential Multiple Markov Chain SA (SMMC-SA) is given in Fig. 5.1. The characteristic equation for the error probability achieved by this algorithm can be derived from Eq. 5.2 and is given by Eq. 5.3. Thus, from Eq. 5.3, for $0 < K/n < 1$, it can be seen that the probability of the error of not getting an optimal solution decreases exponentially as S increases in using SMMC-SA [11]. The computational time increases linearly as the function of S. If the time required to move n iteration by a SSMC-SA is given by t, then a SMMC-SA requires $S \times t$ time to operate every independent simulated annealing by n iteration. Thus, if $S \times t$ is very large for better results, the time consumption can be a problem limiting the use of SMMC-SA. This problem may constraint SMMC-SA to work on a rather small number of Markov chains and/or shorten the run length of each individual Markov chain. However, using fewer

number of Markov chains or shortening the run length of the individual Markov chain can decrease the effectiveness of the algorithm. This limitation can be alleviated using parallel computing by distributing the S Markov chains to multiple computers. Each computer uses a separate pseudo-random number generator to explore different areas of the search space.

```
SMMC-SA()    //Sequential Multiple Markov Chain SA
{
   Randomly generate S solution points {X_{0,1}, X_{0,2}, ···, X_{0,S}}
   Set initial temperature T_0
   Set n = 0, r = 0;
   REPEAT
   {
      FOR( q = 1 to Q )    //Q = number of iteration at temperature T_r
      {
         FOR j = 1 to S    //S = number of independent Markov chains
         {
            Generate X'_{n,j} from X_{n,j}

            IF E(X'_{n,j}) ≤ E(X_{n,j})
               X_{n+1,j} = X'_{n,j}
            ELSE IF exp((E(X_{n,j}) - E(X'_{n,j}))/T_r) > ρ
               X_{n+1,j} = X'_{n,j}
            ELSE
               X_{n+1,j} = X_{n,j}
         }
         n = n + 1
      }
      r = r + 1
      T_r = α × T_{r-1}
   }
   UNTIL a stoping criterion is met
   X_n = X_{n,j*} such that E(X_{n,j*}) ≤ E(X_{n,j}) for j = 1 to S
}
```

Figure 5.1: Psudocode for an instance of a SMMC-SA

$$P(X_n \notin U_{min}) = \prod_{i=1}^{S} P(X_{n,i} \notin U_{min}) \sim \left(\frac{K}{n}\right)^{\theta S} < \left(\frac{K}{n}\right)^{\theta} \tag{5.3}$$

5.2.3. Parallelizing Multiple Markov Chain SA

As discussed in the previous section, SMMC-SA may result in an exponential reduction of the error probability with a linearly increasing computational time as the number of independent simulated annealing runs increases. A promising technique to achieve this exponential reduction of the error probability with less or no increment of computational time can be to use parallel computing. By means of parallelization, it is more likely to have a shortening of the total computation time in proportion to the number of processors used while keeping the total number of iterations unchanged. In order to achieve this, let the S Markov chains be partitioned into equal sub-groups as $S_1, S_2, \cdots, S_p$ and distributed to p concurrently available processors. The computational time for the Parallel Multiple Markov Chain SA (PMMC-SA), t_p will be equal to t_s/p where t_s is the computational time for the SMMC-SA. This computational time may slightly increase if less frequent communication takes among the parallel processors. Parallelizing SMMC-SA may also be done to improve solution quality while keeping the computational time unchanged. Assume SMMC-SA with S Markov chains requiring t_s unit time to perform n iterations in each search direction. The value of S can be increased in many folds keeping t_s and n unchanged using parallel computing to further reduce the error probability. This can be done by having a multiple copies of SMMC-SA to run on several concurrently available computers. A good introduction on parallel simulated annealing with multiple Markov chains can be found in [10], [86], and [100] where the authors assumed only a single Markov chain per processor. Since the exponential reduction of error probability suggests that multiple short SA runs are better than a single long run, in our parallel simulated annealing we have allowed several shorter Markov chains to be traced on each processor.

5.2.4. Components of the Proposed SA

The simulated annealing shares many similar components and features of the genetic algorithm presented in Section 4.2.

Energy Function

The energy function is used to make a stochastic decision whether to accept a new solution or not according to Eq. (5.1). The fitness function of the genetic algorithm given in Eq. (4.2.3) is used as energy function for the simulated annealing in solving the same mathematical model.

Search Space/Solution Representation

The search space determines the number of potential solutions that may be visited by a search heuristics and may influence the performance of the heuristic. The structure of search space is mainly determined by the solution representation adopted. The simulated annealing uses a similar solution representation given in Figure 4.1 as that of the genetic algorithm . The procedure of decoding a solution and the machine assignment heuristics of the GA are also used by the SA.

Neighborhood Generation/Solution Perturbation

Well designed solution perturbation operators are critical to the success of simulated annealing. The proposed SA uses perturbation operators that are similar to the mutation operators of the GA. These are part-level cell assignment perturbation (SP1), operation-level cell assignment perturbation (SP2), subcontract perturbation (SP3), alternative route perturbation (SP4), subcontract degenerator (SP5), and alternative route degenerator (SP6). The perturbation operator SP1 plays the role

of part level cell mutator of the GA and is applied under a probability p_1. Similarly, SP2 plays the role of operation level cell mutator of the GA and is applied with a probability p_2. SP3 randomly steps up or down the variable $\bar{\eta}_i(t)$ with a fixed value under probability p_3. SP4 randomly steps up or down the variable x_j's for all the operations having alternative routings with a fixed value while keeping these values in [0, 1]. This operator is effected with probability p_4. SP5 and SP6 are non-probabilistic operators and are similar to the degenerator operators of the GA. A neighborhood solution violating machine separation constraint is also repaired using the repair heuristic presented in Section 4.2.5.

Two Search Phases

Similar to the GA, the SA run in two search phases by using different solution perturbation operators valid to the different phases of the search. In the first phase, independent cells are formed. In the second phase, the solution found during the first phase is gradually improved to generate cells optimizing inter cell movement and other cost terms of the model.

Cooling Schedule

The rules to determine (a) how high the starting temperature should be, (b) when the current temperature should be lowered, and (c) by how much the temperature should be lowered are called cooling schedule. In the proposed SA, we use a quite popular cooling schedule in which a specified number of iterations are performed at a constant temperature $T_r = \alpha \times T_{r-1}$. In this cooling schedule, r is the index of temperature levels and the constant α, in $[0, 1]$, is the cooling coefficient and generally chosen to be close to 1. Thus, the parameters of the proposed simulated annealing method associated with the cooling schedule are the initial temperature

T_0, the cooling exponent α, and the number of iteration at each temperature level.

Interactions of Markov Chains

In the proposed simulated annealing algorithm, S Markov chains are followed by each of the P concurrently available processors. For better performance, these Markov chains may be allowed to interact within and across the processors. In this study we consider two interaction schemes of the Markov chains. In the first scheme, the master computer periodically gathers the best solution so far found by each computer and determines a winner solution as the best of all. Once the master computer determines the winner solution, it broadcast this winner solution to all the computers. Then, at this same period, each computer will start all the Markov chains in its domain from the winner solution at current level of the temperature. In the second scheme, similar to the first one, the master and the worker computers interact periodically to determine the winner solution. However, each computer will dynamically determine the number of successive iterations each Markov chain may elapse without improvement and update those that do not improve for a certain large number of consecutive iterations. Thus the computers are interacting periodically while each computer update its Markov chains dynamically when necessary rather than with a fixed period.

5.2.5. Steps of the PSA

The steps of the parallel simulated annealing based on the first interaction scheme of the Markov chains are presented next. Several symbols are used in presenting the algorithm which are explained below.

p Processor Index (processor ID), $p = 0,\ 2,\ ...,\ P-1$ where P is the number

of concurrently available processors

s Index of Markov chain, $s = 1,\ 2,\ ...,\ S$ where S is the number of Markov chains followed by each processors

n Iteration counter, $n = 1,\ 2,\ ...,\ N$ where N is the maximum number of iterations in each Markov chain

X_{nsp} The solution at the n^{th} iteration along the s^{th} Markov chain in the p^{th} processor

α Cooling schedule exponent

r Index for the temperature levels in the cooling schedule

T_r Temperature at the r^{th} level, $T_r = \alpha \times T_{r-1} = \alpha^r \times T_0$

Q Number of iterations to be performed in each Markov chain at each temperature level

CF Communication frequency. It is defined as the number of iterations to be performed by each Markov chain before communication is effected among the processors

BI_p Best feasible individual so far found in the p^{th} processor

BI Best feasible individual so far found by all the processors

$Phase$ The current phase of the search which equals to 1 for the first phase or 2 for the second phase

C_{phase} The number of iterations to be performed by each Markov chain before the search phase is changed from $Phase = 1$ to $Phase = 2$

rand() Random number generator

Using the above notations, the steps of the proposed simulated annealing based heuristic are listed below. These steps were coded in C++ with MPI message-passing library for communication. The code was tested in a parallel computation environment composed of several waiters each containing a Pentium 4 processor (3.2 GHz, 2GB RAM).

Step 0. Initialization

Set p = My_Processor_ID

Initialize counter: $n = 0$, $r = 0$, and $Phase = 1$. Initialize the Best-Individual BI_p with a null value. If $p = 0$, set the Best-Individual BI so far found with a null value too.

Randomly generate initial solution points $X_{0,1,p}$, $X_{0,2,p}$, $\cdots$ $X_{0,S,p}$ such that each of them represents a solution having independent cells (i.e. the c_j's of all the operations of a given part are set equal).

Repair initial solutions violating machine separation constraint using the Repair Heuristic.

For $s = 1$ to S : Decode $X_{0,sp}$ for the continuous variable and determine the corresponding values of the integer variable using the Machine Assignment Heuristic and then evaluate $E(X_{0,sp})$.

Step 1. For $s = 1$ to S

Move: Using the perturbation operators valid to the current phase of the search, perturb X_{nsp} to get X'_{nsp}. If X'_{nsp} violates machine separation constraint, repair it using Repair Heuristic.

Evaluate: Decode X'_{nsp} for the continuous variable and determine the corresponding values of the integer variable using the Machine Assignment Heuristic. Evaluate $E(X'_{nsp})$.

Decide: **If** $E(X'_{nsp}) \leq E(X_{nsp})$, **then** $X_{n+1,sp} = X'_{nsp}$
Else If $\exp\left\{\left[E(X'_{nsp}) - E(X_{nsp})\right]/T_r\right\} >$ rand(), **then** $X_{n+1,sp} =$

X'_{nsp}

Else $X_{n+1,sp} = X_{nsp}$

Update: **If** $E(X_{n+1,sp}) < E(BI_p)$, **then** $BI_p = X_{n+1,sp}$

Step 2. Set $n = n + 1$.

If $n = C_{phase}$, then set $Phase = 2$.

If n mod $Q = 0$, then set $r = r + 1$, and $T_r = \alpha \times T_{r-1}$

If $(n < N + 1)$ AND $(n \text{ mod } CF \neq 0)$ go to Step 1.

If $(n \not< N + 1)$ AND $(n \text{ mod } CF = 0)$ go to Step 3.

If $n = N + 1$, STOP.

Step 3. If $p \neq 0$, send BI_p to the processor whose Processor_ID = 0.

If $p = 0$, receive the best solutions found by each processor and determine BI and send this solution to all the other processors.

If $p \neq 0$, receive BI from the processor whose Processor_ID = 0.

Set $X_{nsp} = BI$ for all s and go to Step 1.

5.3. Computational Performances of the PSA

Using the problem considered in evaluating the genetic algorithm, the performance of the developed parallel simulated annealing is evaluated for 15 different parameter settings to test its robustness. These test parameters are given in Table 5.1. The first two parameters in the fist group are the scale factors for the penalty terms in the energy function as explained in Section 4.2.3. The third parameter

Table 5.1: Parameter settings for 15 different test cases on Example 1

Test No.	Parameters related to: Energy Function f_{wb}	f_{cs}	f_{sub}	Perturbation Operators p_1	p_2	p_3	p_4	Cooling Schedule Q	T_0	α
1	1.000	100	200	0.010	0.010	0.050	0.0050	120000	10000000	0.985
2	0.005	100	300	0.030	0.020	0.060	0.0010	60000	12000000	0.900
3	0.020	200	400	0.040	0.060	0.030	0.0080	30000	900000	0.985
4	2.000	250	150	0.250	0.100	0.100	0.0100	60000	10000000	0.850
5	0.010	400	250	0.230	0.150	0.100	0.0150	70000	16000000	0.885
6	0.003	150	150	0.350	0.250	0.150	0.0070	50000	1000000	0.985
7	1.500	200	150	0.050	0.040	0.050	0.1000	40000	10000000	0.985
8	0.600	200	150	0.150	0.240	0.050	0.0010	90000	12000000	0.985
9	0.200	300	100	0.350	0.150	0.050	0.0050	80000	20000000	0.755
10	0.250	100	100	0.150	0.250	0.050	0.0150	100000	10000000	0.985
11	1.500	400	150	0.055	0.040	0.055	0.1800	90000	10000000	0.985
12	0.055	100	300	0.035	0.320	0.066	0.0015	30000	9000000	0.885
13	0.150	200	220	0.355	0.220	0.008	0.0150	90000	12000000	0.985
14	0.250	100	220	0.300	0.250	0.018	0.0150	70000	7000000	0.985
15	0.003	150	250	0.250	0.150	0.100	0.0070	90000	15000000	0.955

f_{sub} is introduced here as a scale factor for the penalty term of subcontracting parts as we assumed no part shall be subcontracted for this numerical example. The second group of parameters are the probabilities by which the four probabilistic perturbation operators are applied as explained in Section 5.2.4. The last group of parameters are related to the cooling schedule and these are the number of iterations Q of the SSMC-SA at each temperature level, the initial temperature T_0, and the cooling exponent α.

We implemented the PMMC-SA on 12 connected PCs within the RQCHP cluster in Montreal. In each computer 30 Markov chains are simultaneously traced. The cooling schedule consists of a sequence of 2000 temperature levels. The number of iterations at each temperature level in each Markov chain is set to Q/30. The communication among the computers is set to take place every 100,000 iterations. A Markov chain that do not improve for consecutive 300,000 SA moves re-starts

Table 5.2: Comparison of the proposed parallel SA with LINGO

Time	LINGO		The proposed Parallel SA				
hr:min:sec	Lower Bound	Objective	Test 1	Test 2	Test 3	Test 4	Test 5
00:00:15	-	×	1.86206	1.94255	1.78819	1.90043	2.07089
00:00:30	1.69274	×	1.78606	1.77391	1.78810	1.81659	1.84362
00:01:00	1.69274	×	1.78606	1.77150	1.78695	1.79369	1.84362
00:02:00	1.69274	×	1.78606	1.77124	1.77153	1.79236	1.84362
00:06:00	1.69274	1.86962	1.78606	1.77023	1.77153	1.79207	1.84362
00:08:00	1.69275	1.86962	1.78606	1.77023	1.76291	1.79190	1.76568
00:10:00	1.69277	1.86962	1.78210	1.77023	1.76291	1.79185	1.76330
00:12:00	1.69282	1.81068	1.77932	1.77023	1.76067	1.79174	1.76317
00:16:00	1.69282	1.81068	1.77774	1.76536	1.75881	1.79140	1.76287
00:20:00	1.69309	1.81068	1.77774	1.76518	1.75844	1.79140	1.76245
00:30:00	1.69311	1.77401	1.77705	1.76223	1.75752	1.76378	1.76209
01:00:00	1.69323	1.76934	1.76628	1.76054	1.75746	1.76060	1.75927
01:30:00	1.69323	1.76934	1.76338	1.75990	*	1.75933	1.75795
02:00:00	1.69323	1.76934	1.76180	*	*	1.75907	1.75792
02:30:00	1.69323	1.76934	1.76081	*	*	*	*
03:00:00	1.69344	1.76934	1.76081	*	*	*	*
05:00:00	1.69346	1.76865	*	*	*	*	*
10:00:00	1.69353	1.76865	*	*	*	*	*
15:00:00	1.69366	1.76865	*	*	*	*	*
20:30:00	1.69377	1.76865	*	*	*	*	*

× - Feasible solution was not found
* - A termination criterion was met.
Note: Values are in Millions.

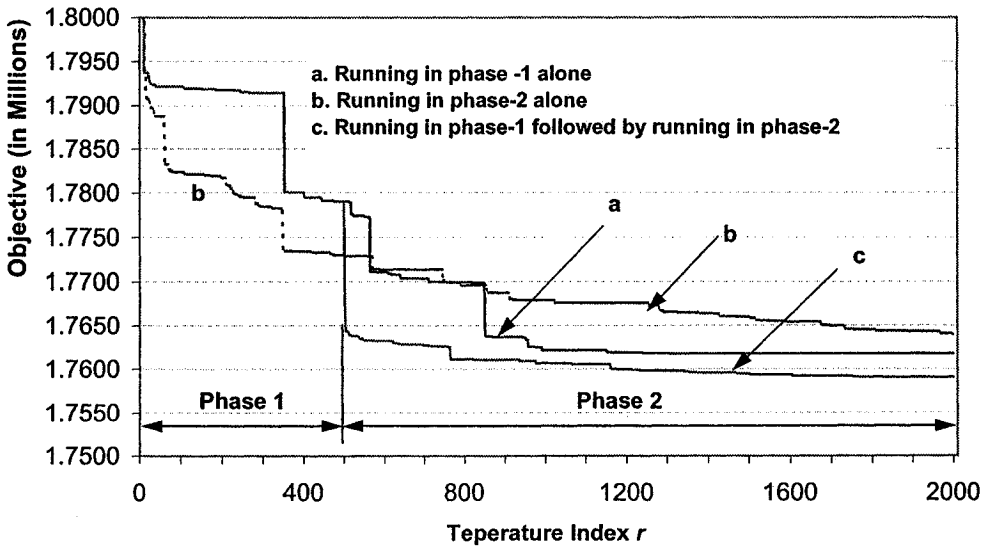


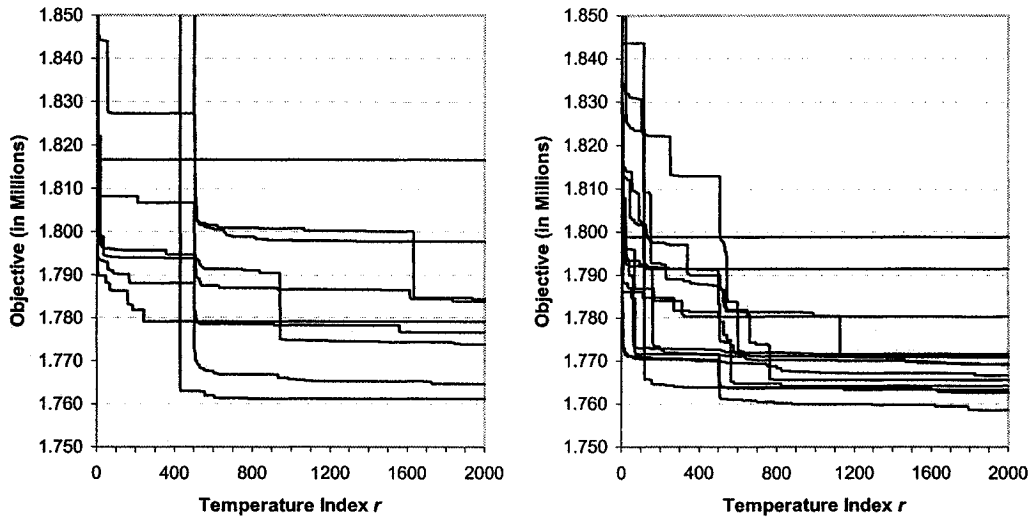
Figure 5.2: The impact of dividing the search into pases

Table 5.3: Objective function value of the best solution found in 15 test cases

Test			PMMC-SA (Type of Interaction scheme)		
No.	SSMC-SA	SMMC-SA	Non-Interacting	First Scheme	Second Scheme
1	1.77660	1.76410	1.76240	1.77689	1.76081
2	1.81660	1.76321	1.75741	1.76051	1.75990
3	NFS	1.75851	1.75851	1.75839	1.75746
4	1.78434	1.76919	1.76099	1.75795	1.75907
5	NFS	1.76252	1.76014	1.75732	1.75792
6	NFS	NFS	1.78415	1.76269	1.76515
7	1.86653	1.78025	1.77973	1.78171	1.78171
8	1.78389	1.77081	1.76448	1.76327	1.76134
9	1.86196	1.76927	1.76211	1.75865	1.76059
10	1.76458	1.76658	1.76153	1.75874	1.76079
11	1.77913	1.79145	1.77208	1.77099	1.77099
12	1.76120	1.77087	1.76248	1.75976	1.76057
13	1.77382	1.77157	1.76677	1.76346	1.76491
14	1.79781	1.76556	1.76556	1.76571	1.76128
15	NFS	1.79873	1.76038	1.75798	1.75778
BTL	2	5	12	12	13

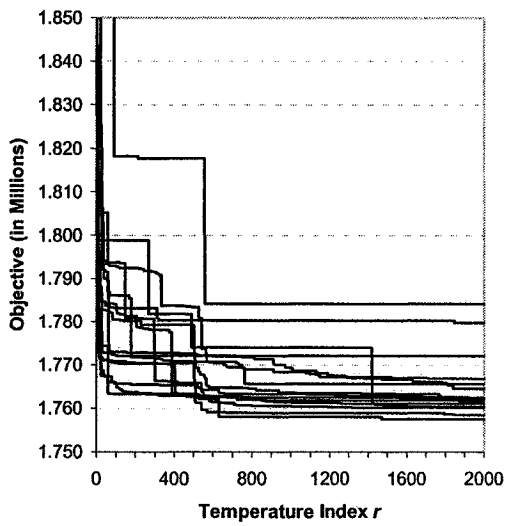
NFS - No Feasible Solution was found
BTL - Number of times a solution Better Than LINGO is found

from the best known solution at the current temperature level. The performance of this parallel simulated annealing was evaluated against the results from LINGO. To find solutions using LINGO, the absolute value term of the objective function in the integer programming model was linearized. As can be seen from Table 4.3, the lower bound and the best objective function value found by LINGO after several hours of computation were 1.69346 and 1.76865 respectively. From this table, it can be seen that, in most of the test cases, the developed heuristic was able to determine solutions that LINGO could not find after several hours of computation. Such results were achieved by using those problem specific procedures as systematically dividing search phases and the related perturbation operators, constraint handling and parallel computing. As shown in Figures 4.7, dividing the search into phases enables the parallel SA algorithm to find near optimal solutions quickly.

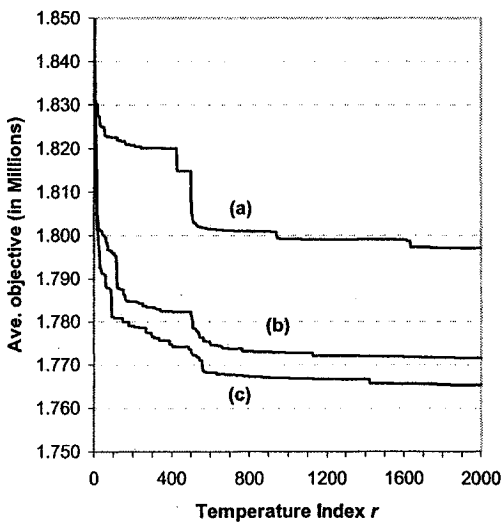


(a) SSMC-SA

(b) SMMC-SA



(c) PMMC-SA: Non-Interacting



(d) Average Convergence

Figure 5.3: Convergence graphs of (a) SSMC-SA, (b) SMMC-SA, (c) PMMC-SA non-interaction (d) Average convergence of a, b, and c for the 15 test cases

5.3.1. Single Versus Multiple Markov Chain SA

In this section, SSMC-SA and SMMC-SA were compared for their performance by keeping the total number of SA iterations the same in both cases. If the number of iterations performed by the SSMC-SA at a given temperature level is set to Q , then a SMMC-SA with S Markov chains is allowed to execute each of its Markov chains for Q/S iterations at each temperature level. If N levels were used, the total number of iterations in both cases will be the same and equal to $Q \times N$. For this experimental study we set $N = 2000$, $S = 30$ and Q to the values shown in Table 5.1. As shown in the first two columns of Table 5.3, the SMMC-SA reaches better results than that SSMC-SA in 11 test cases. Moreover, SMMC-SA found better results than those from LINGO for 5 times while SSMC-SA found better results twice. These results are further illustrated in Figures 5.3-(a), 5.3-(b) and 5.3-(d). From this comparative study, it is evident that the convergence behavior and the robustness of the algorithm are greatly improved by using multiple short runs instead of one single long run with the same total computational time. This result is further improved by parallel computing as it is shown in Table 5.3 and Figures 5.3-(c) and 5.3-(d). In the parallel computing, multiple copies of SMMC-SA were run on concurrently available computers. By allowing the computers to communicate with each will likely to improve the computational results. Table 5.3 shows that in most cases the solutions found by allowing the Markov chains to interact are better than those found using non-interacting scheme. It also shows that it is more likely to obtain better solutions using the first communication scheme than using the second one. However, from the average convergence graphs given in Figure 5.4, the second interaction scheme is slightly better than the first one. This entails that both interaction schemes may be considered to reduce computational

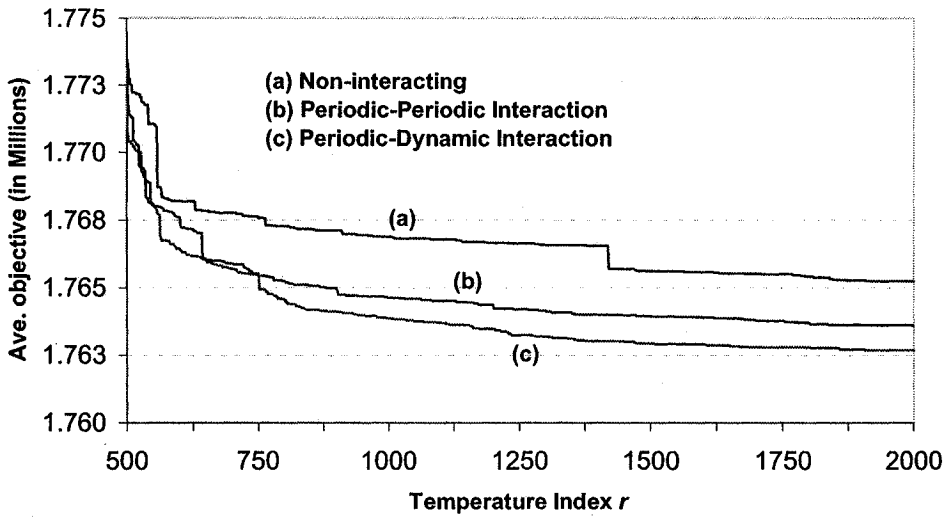


Figure 5.4: The impact of communication schemes on convergence

time in searching for better solutions.

5.4. Summary and Discussions

Simulated annealing is a general-purpose optimization technique capable of finding optimal or near-optimal solutions in various applications. The major disadvantage of this technique is its slow convergence making it not suitable for solving many complex optimization problems. Most SA schemes in literature follow a single Markov chain. However, following a single Markov chain may not be necessary from a performance point of view. Hence, we developed a multiple Markov chain simulated annealing algorithm which allows multiple search directions to be traced simultaneously. The algorithm incorporates several problem specific perturbation operators and constraint handling techniques. Our computational results showed that the multiple Markov chain SA is much more efficient than the conventional single Markov chain SA. The solution quality obtained using such multiple Markov chain SA can be further improved by increasing the number of Markov chains to

be traced. This results in an exponential reduction of the error probability and a linearly increasing computational time. A promising technique to achieve this exponential reduction of the error probability with less or no increment of computational time is parallel computing. To this end, we implement the proposed multiple Markov chain SA in a parallel computing environment. The parallel implementation improved computational efficiency of the multiple Markov chain SA in terms of solution quality and computational time. The parallelization was achieved by distributing the Markov chains to concurrently available processors.

Chapter 6

Mathematical Model-B

6.1. Introduction

Various manufacturing production planning and inventory control problems have been studied extensively by many production management researchers. Different models and methods developed to solve these problems can be found in widely used textbooks of production engineering or manufacturing systems [127, 134]. Inventory control models from simple EOQ to more complicated MRP, kanban and CONWIP [103, 138] models have been developed and widely used in today's manufacturing industry. Some of them are very successful in practical applications. Mathematical programming is also a powerful tool for solving complicated production planning problems, when product structures with multi-item multi-level lot sizing are considered. Review and discussion on mathematical programming models for certain kanban and MRP systems can be found in [119]. Other mathematical programming models for MRP- or kanban-based production planning have also been developed in [13, 15, 120]. Many of the mathematical models and solution methods were developed to solve problems in general manufacturing or service industries and can

be widely applied. Prominent manufacturing features such as production flexibility and manufacturing cell formation were usually not considered in developing production planning models [30]. On the other hand, manufacturing systems analysis tend to study more specific system characters such as job sequencing and scheduling, alternative process plans and different ways of forming manufacturing cells as well as handling production materials and tools (e.g. [17, 32, 64]). As pointed out in Arvindh and Irani [5], an integrated approach should be pursued in manufacturing system analysis, since different aspects of a system are interrelated in many ways. In addition, comprehensive model consisting of different aspects of the system can help one to understand the problem better. Integrated systems approaches can minimize the possibility of certain important aspects of the system being overlooked, while other issues are being studied. Based on the above considerations, we proposed a mathematical model for an integrated approach to study cellular manufacturing and a multi-item multi-level capacitated lot sizing problem. In fact this model is an extension of the model discussed in Chen [30] where product structure (bill-of-material), machine capacity, workload balancing, alternative routings and impacts of lot sizes on product quality were not taken into account.

6.2. The Effect of Run Length on Product Quality

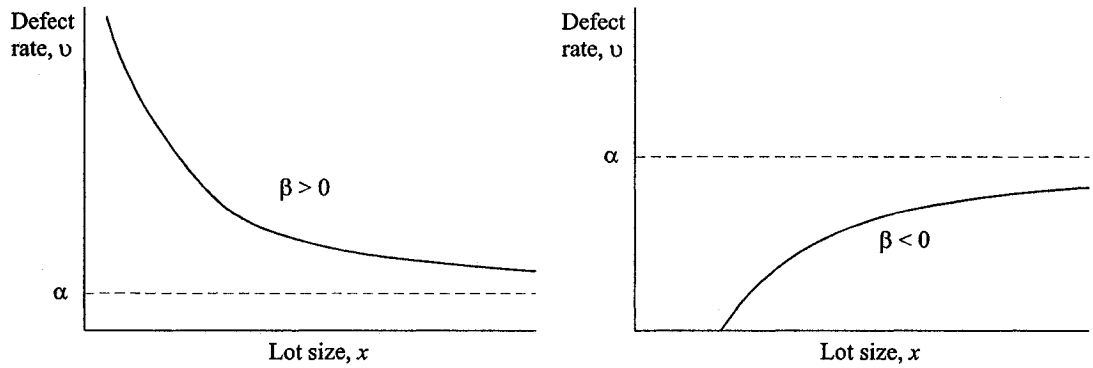
Manufacturers often do not meet their annual quantities through one long production run. Rather, they produce their products in smaller volumes and separate lots at different times of the year. The precise size and timing of the runs is a scheduling decision, which is governed by sales and inventory targets. Quality seldom entered into the equation. Yet at the same time, several managers pointed to a connection between run length and defect rates [44]. Long runs, they noted,

providing a stable environment–the opportunity to master required skill through repetition (Disruptive-Philosophy). As operators become familiar with the production process, defect rates normally fall, and eventually diminishing to a minimum when technical limits are reached and opportunities for learning are exhausted. The worst manufacturing environment is one with short runs and frequent changeovers. Each changeover is likely to cause defects to rise abnormally for a brief period. Moreover, defect rates will fall only a small amount between changeovers as the workers never reap the full benefits of experience due to changing work assignments and short production runs. Garvin [44] pointed out that this observation was strongly supported by statistical analysis and concluded that the differences in run length alone explained 50-70% of the variation in defect rates.

Contrary to the above observation, Huge and Anderson [57], Jaber and Bonney [62], Li and Cheng [89] and Porteus [118] showed that product quality is positively affected by reduced lot-sizes (JIT-Philosophy). Huge and Anderson [57] state that without even working on quality improvement, reject rates improve proportionally with the reduction in lot sizes. *"Prompt identification of defects"* is the major reason frequently given for justifying this positive relationship between short run length and product quality. Smaller lots get used up sooner; hence defective parts are caught earlier [128]. This reduces scrap and rework and allows sources of problems to be quickly caught and corrected. Thus there is an incentive to produce smaller lots, and have a smaller fraction of defective units.

Urban [147] proposed a simple relation between lot size x and defect rate v given by the equation $v = \alpha + \beta/x$. In his model, the number of defects items in a lot size of x is given by vx which is equal to $\alpha x + \beta$. The constant α is chosen in $[0, 1]$. The other constant β is positive for JIT-philosophy and negative for the disruptive philosophy. The plots of v versus x are given in Figure 6.1. His objective

is to incorporate the influence of run length on product quality into a production planning model that can be utilized under either JIT- or disruptive philosophy. He did not justify or condone one philosophy over the other.



(a) Disruptive-Philosophy (b) JIT-Philosophy

Figure 6.1: The impact of lot size on product quality (Urban [147])

6.3. Lot Sizing by Considering Product Quality

As it was indicated in Section 6.2, several authors noted the impact of run length on product quality. Jaber and Bonney [62], Li and Cheng [89] and Porteus [118] presented an economic production quantity (EPQ) model for a single item following the JIT-Philosophy. Urban [147] developed a single item EPQ model which allowing the user to follow either Disruptive- or JIT-Philosophy. In this section we extend the approach of Urban [147] by developing a model to address a multi-item multi-level capacitated lot sizing problem (MLCLSP). We use this model to demonstrate the impact of run length on product quality and as a bases for an integrated lot sizing and cell formation model presented in the next section. The notations and the mathematical model of the MLCLSP are presented below.

Indexes:

t - Period index: $t = 1, 2, \cdots, T$

i - Part index: $i = 1, 2, \cdots, I$

r - Route index of part i: $r = 1, 2, \cdots, R_i$

j - Index of operations of part i in route r: $j = 1, 2, \cdots, O_{ri}$

k - Machine index: $k = 1, 2, \cdots, K$

Input Data:

$d_i(t)$ - Demand for part i in time period t

Γ_i - The set of immediate successor items to item i

$r_{ii'}$ - The number of item i needed by one unit of item i', where $i' \in \Gamma(i)$

m_{jri} - Index of the machine type used to process operation j of part i along route r

λ_{jri} - Processing time in minutes of operation j of part i along route r

S_{ri} - Cost to set up route r of part i

$\Re_i$ - Average rework and replacement cost of one defective item of type i

H_i - Unit inventory carrying cost per period for item i

P_k - Procurement cost of type k machine,

G_k - Maintenance and other overhead cost of machine type k,

O_k - Operation cost per hour of type k machine

C_k - Capacity of one unit of type k machine

R_k^+ - Cost of installing one unit of type k machine,

M - Large positive number,

Decision Variables:

Continuous:

$x_{ri}(t)$ - The production sub-lot size of item i along route r in period t

$\bar{x}_i(t)$ - The quantity of item i outsourced in period t through subcontracting

$I_i(t)$ - Inventory level of product i at the beginning of time period t

General Integer:

$N_k(t)$ - Number of type k machines in the system in period t

$y_k^+(t)$ - Number of type k machines procured at the beginning of period t

Binary:

$z_{ri}(t)$ - $\begin{cases} 1, & \text{if route } r \text{ of part } i \text{ is set up for production during time period } t \\ 0, & \text{otherwise.} \end{cases}$

Minimize:

$$\begin{aligned} Z &= \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K} N_k(t)\cdot G_k \\ &+ \sum_{k=1}^{K} P_k \cdot y_k^+(t) \\ &+ \sum_{t=1}^{T}\sum_{i=1}^{P}\sum_{r=1}^{R_i}\sum_{j=1}^{O_{ri}} x_{ri}(t)\cdot\lambda_{jri}\cdot O_{m_{jri}} \\ &+ \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K} R_k^+ \cdot y_k^+(t) \\ &+ \sum_{t=1}^{T}\sum_{i=1}^{P} \Phi_i \cdot \bar{x}_i(t) \\ &+ \sum_{t=1}^{T}\sum_{i=1}^{P}\sum_{r=1}^{R_i} S_{ri}\cdot z_{ri}(t) \\ &+ \sum_{t=1}^{T}\sum_{i=1}^{P} H_i \cdot I_i(t) \\ &+ \sum_{t=1}^{T}\sum_{i=1}^{P} \Re_i \cdot \sum_{r=1}^{R_i}[\alpha_{ri}\cdot x_{ri} + \beta_{ri}\cdot z_{ri}(t)] \end{aligned} \tag{6.1}$$

Subject to:

$$x_{ri}(t) \le M \cdot z_{ri}(t) \quad ; \quad \forall(r,i,t) \tag{6.2}$$

$$I_i(t-1) + \sum_{r=1}^{R_i} x_{ri}(t) + \bar{x}_i(t) - \sum_{r=1}^{R_i}[\alpha_{ri} \cdot x_{ri} + \beta_{ri} \cdot z_{ri}(t)]$$

$$-I_i(t) = d_i(t) + \sum_{j \in \Gamma_i} r_{ij} \cdot \left(\sum_{r=1}^{R_j} x_{rj}(t) + \bar{x}_j(t) - \right.$$

$$\left. \sum_{r}^{R_j}[\alpha_{rj} \cdot x_{rj} + \beta_{ri} \cdot z_{rj}(t)] \right) \quad ; \quad \forall(i,t) \qquad (6.3)$$

$$\alpha_{ri} \cdot x_{ri} + \beta_{ri} \cdot z_{ri}(t) \geq 0 \quad ; \quad \forall(r,i,t) \qquad (6.4)$$

$$\sum_{\forall(j,r,i)|m_{jri}=k} x_i(t) \cdot z_{ri}(t) \cdot \lambda_{jii} \leq C_k \cdot N_k(t) \quad ; \quad \forall(k,l,t) \qquad (6.5)$$

$$N_k(t) = N_k(t-1) + y_k^+(t) \quad ; \quad \forall(k,t) \qquad (6.6)$$

$$N_k(t) - N_k(t-1) \geq 0 \quad ; \quad \forall(k,t)|t > 1 \qquad (6.7)$$

$$N_k(t) \in \{0,1,2,\cdots\} \quad ; \quad \forall(k,t) \qquad (6.8)$$

$$z_{ril}(t) \in \{0,1\} \quad ; \quad \forall(r,i,t) \qquad (6.9)$$

In the above model we assume that there are a total of P part types, and some of which are end items with known external demand in each of the T periods of the planning horizon. We further assume that the component items that appear in the product structures can also have a know external demand in each period in addition to the dependent demand generated by the successor items. The product structures which specify the assembly relations are assumed to be general in the sense that Γ_i may have more than one element; on the other hand, the product structures are called linear or assembly product structures if Γ_i has a single element for all the items. A part may be processed in different alternative routes and an entire lot of a part may split into these different alternative routes. In each route a part may requires several operations to be processed on different machine types. Accordingly the machining times needed on each machine type by each operation are known. Part production may be subcontracted for economic reasons. Capacity limitations are imposed on the availabilities of the machines. If additional machines

are required to meet capacity requirement, these machines can be procured provided that there is an economic advantage over part subcontracting.

The objective function given in Eq. (6.1) comprises seven cost terms. These cost terms are related to (1) machine maintenance and holding, (2) machine procurement, (3) machine operation, (4) installing new machines (5) part sub-contracting, (6) setup, (7) inventory holding, and (8) replacement of defective parts. The constraint in Eq. (6.2) ensures that setup is performed (i.e. $z_{ir}(t) = 1$) whenever there is production in period t (i.e. $x_{ir}(t) > 0$). Eq. (6.3) is inventory balance constraint. It states that for each item in each period the beginning inventories together with the production quantity less the defect allowance and the inventory at the end of the period should meet the external demand and the dependent demand generated by the non-defective successor items. The constraint in Eq. (6.4) enforces non-negativity on the number of defective items under the JIT-Philosophy in which β_i is negative. This constraint is not required if the decision maker applies the Disruptive-Philosophy. The capacity limitations of the machines are expressed in Eq. 6.5. Eq. (6.6) states that the number of type k machines in the system in period t is equal to the number of machines in period $t-1$ plus those procured and installed in period t. Eq. (6.7) states that machines will not be removed from the system. The constraints in Eq. (6.8) and (6.9) are integrality constraints.

6.4. Integrated Cell Formation and Lot Sizing

In this section we present the problem description, the mathematical formulation and the features of the proposed integrated cell formation and MRP based lot sizing model. The choice of MRP as the production planning technique in the integrated model follows the recent report in [58] which states most CMS user firms

employ MRP in their production planning activities. The MLCLSP model presented in the previous section is used as the bases for the production planning part of the integrated cell formation and lot sizing model.

6.4.1. Problem Description

Consider a manufacturing system consisting of a number of machines to process different parts. A part may be processed in different alternative routes and in each route it may requires several operations to be processed on different machine types in a given sequence. An entire lot of a part may be split into different alternative routes. In addition, we consider the manufacturing system in a number of time periods. One time period could be a month, a season, or a year. Each machine has a limited capacity expressed in hours during each time period. Machines can have one or more identical copies to meet capacity requirements and reduce/eliminate inter-cell movement. Bill-of-material of finished products is known. The independent demand for the parts vary with t in a deterministic manner. In planning the production, the production volume of a part in a given period should be deducted from the number of finished parts in storage, the level of its independent demand and the required production volume of the parent parts in the products structure during that same period. Moreover, the effect of the run length of production (lot size) on product quality will be considered in account of either JIT- or disruptive-philosophy. The problem is to group the machines into relatively independent cells with minimum inter-cell movement of the parts and decide on the subsequent system reconfiguration and lot sizes of the parts in order to minimize the system operation costs. In grouping the machines, it is also required that the workload of the cells should be balanced. Machines that cannot be located in the same cell should be

separated. Machines that cannot be separated should be co-located. The overall objective is to minimize the system costs due to machine maintenance and overhead, machine procurement, inter-cell materials handling, machine operation, system re-configuration, part subcontracting, setup, inventory holding, and replacement of defective parts for the entire planning horizon.

Additional Indexes:

l - Cell index: $l = 1, 2, \cdots, L$

Additional Input Data:

V_i - Unit cost to move part i between cells,

Θ - A set of machine pairs $\{(k^a, k^b)/k^a, k^b \in \{1, \cdots, K\}, k^a \neq k^b$, and k^a cannot be placed in the same cell with $k^b\}$

LB_l - Minimum number of machines in cell l

UB_l - Maximum number of machines in cell l

R_k^- - Cost of removing one unit of type k machine

Decision Variables:

General Integer:

$N_{kl}(t)$ - Number of type k machines assigned to cell l at the beginning of period t

$y_{kl}^+(t)$ - Number of type k machines added to cell l at the beginning of period t

$y_{kl}^-(t)$ - Number of type k machines removed from cell l at the beginning of period t

Continuous:

$x_{ri}(t)$ - The production sub-lot size of item i along route r in period t

$\bar{x}_i(t)$ - The quantity of item i outsourced in period t through subcontracting

$I_i(t)$ - Inventory level of product i at the beginning of time period t

Binary:

$r_{kl}(t)$ - $\begin{cases} 1, & \text{if type } k \text{ machines are to be assigned to cell } l \text{ during time period } t \\ 0, & \text{otherwise.} \end{cases}$

$z_{ri}(t)$ - $\begin{cases} 1, & \text{if route } r \text{ of part } i \text{ is set up for production during time period } t \\ 0, & \text{otherwise.} \end{cases}$

$\eta_{jril}(t)$ - $\begin{cases} 1, & \text{if } j^{th} \text{ operation in route } r \text{ of part } i \text{ is processed in cell } l \text{ during time period } t \\ 0, & \text{otherwise.} \end{cases}$

6.4.2. Objective Function and Constraints

Following the problem description and notations given in the previous section, the integrated mathematical model for cellular manufacturing system design and production planning is presented below.

Minimize:

$$
\begin{aligned}
Z \;=\; & \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K} N_{kl}(t)\cdot G_k \\
& + \sum_{k=1}^{K} P_k \cdot \left(\sum_{l=1}^{L} N_{kl}(T) - \sum_{l=1}^{L} N_{kl}(0)\right) \\
& + \frac{1}{2}\sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{i=1}^{P}\sum_{r=1}^{R_i}\sum_{j=1}^{O_{ri}-1} V_i \cdot x_{ri}(t)\,\left|\eta_{j+1,ril}(t) - \eta_{jril}(t)\right| \\
& + \sum_{t=1}^{T}\sum_{i=1}^{P}\sum_{r=1}^{R_i}\sum_{j=1}^{O_{ri}} x_{ri}(t)\cdot \lambda_{jri}\cdot O_{m_{jri}} \\
& + \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K}\left(R_k^{+}\cdot y_{kl}^{+}(t) + R_k^{-}\cdot y_{kl}^{-}(t)\right)
\end{aligned}
$$

$$
\begin{aligned}
&+ \sum_{t=1}^{T}\sum_{i=1}^{P} \Phi_i \cdot \bar{x}_i(t) \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{P}\sum_{r=1}^{R_i} S_{ri} \cdot z_{ri}(t) \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{P} H_i \cdot I_i(t) \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{P} \Re_i \cdot \sum_{r=1}^{R_i}[\alpha_{ri} \cdot x_{ri} + \beta_{ri} \cdot z_{ri}(t)]
\end{aligned} \tag{6.10}
$$

Subject to:

$$\sum_{l=1}^{L} \eta_{jril}(t) = z_{ri}(t) \quad ; \quad \forall(j,r,i,t) \tag{6.11}$$

$$x_{ri}(t) \leq M \cdot z_{ri}(t) \quad ; \quad \forall(r,i,t) \tag{6.12}$$

$$
\begin{aligned}
I_i(t-1) + \sum_{r=1}^{R_i} x_{ri}(t) + \bar{x}_i(t) - \sum_{r=1}^{R_i}[\alpha_{ri} \cdot x_{ri}(t) + \beta_{ri} \cdot z_{ri}(t)] \\
-I_i(t) = d_i(t) + \sum_{j \in \Gamma_i} r_{ij} \cdot \left(\sum_{r=1}^{R_j} x_{rj}(t) + \bar{x}_j(t) - \right. \\
\left. \sum_{r}^{R_j}[\alpha_{rj} \cdot x_{rj}(t) + \beta_{ri} \cdot z_{rj}(t)] \right) \quad ; \quad \forall(i,t)
\end{aligned} \tag{6.13}
$$

$$\alpha_{ri} \cdot x_{ri} + \beta_{ri} \cdot z_{ri}(t) \geq 0 \quad ; \quad \forall(r,i,t) \tag{6.14}$$

$$\sum_{\forall(j,r,i)|m_{jri}=k} x_i(t) \cdot \eta_{jril}(t) \cdot \lambda_{jii} \leq C_k \cdot N_{kl}(t) \quad ; \quad \forall(k,l,t) \tag{6.15}$$

$$\sum_{l=1}^{L} N_{kl}(t) - \sum_{l=1}^{L} N_{kl}(t-1) \geq 0 \quad ; \quad \forall(k,t) \tag{6.16}$$

$$
\begin{aligned}
\sum_{i=1}^{P}\sum_{r=1}^{R_i}\sum_{j=1}^{O_{ri}} x_{ri}(t) \cdot \eta_{jril}(t) \cdot \lambda_{jri} \geq \\
\frac{q}{L}\sum_{l=1}^{L}\sum_{i=1}^{P}\sum_{r=1}^{R_i}\sum_{j=1}^{O_{ri}} x_{ri}(t) \cdot \eta_{jril}(t) \cdot \lambda_{jri} \quad ; \quad \forall(l,t)
\end{aligned} \tag{6.17}
$$

$$N_{kl}(t) = N_{kl}(t-1) + y^{+}_{kl}(t) - y^{-}_{kl}(t) \quad ; \quad \forall(k,l,t) \tag{6.18}$$

$$LB_l \leq \sum_{k=1}^{K} N_{kl}(t) \leq UB_l \quad ; \quad \forall(l,t) \tag{6.19}$$

$$N_{kl}(t) \leq M^{\infty} \cdot r_{kl}(t) \quad ; \quad \forall(k,l,t) \tag{6.20}$$

$$r_{kl}(t) \leq N_{kl}(t) \quad ; \quad \forall(k,l,t) \qquad (6.21)$$

$$r_{k^a l}(t) + r_{k^b l}(t) \leq 1 \quad ; \quad \left(k^a, k^b\right) \in \Theta, \quad \forall(l,t) \qquad (6.22)$$

$$r_{k^c l}(t) - r_{k^d l}(t) = 0 \quad ; \quad \left(k^c, k^d\right) \in \Omega, \quad \forall(l,t) \qquad (6.23)$$

$$y^+_{kl}(t), \quad y^-_{kl}(t), \quad N_{kl}(t) \in \{0,1,2,\cdots\} \quad ; \quad \forall(k,l,t) \qquad (6.24)$$

$$\eta_{jril}(t), z_{ri}(t), \quad r_{kl}(t) \in \{0,1\} \quad ; \quad \forall(j,r,i,l,t) \qquad (6.25)$$

Model Objective Function: The objective function given in Eq. (6.10) comprises several cost terms. The first term is machine maintenance and overhead costs. The second term is machine procurement cost at the beginning of each period. In this cost term, $N_{kl}(0)$ stands for the number of type k machines available from the previous job shop system. $N_{kl}(0) = 0, \forall k$ in the case of setting up a new system. The third term of the objective function represents the inter-cell material handling cost. The forth, fifth, and sixth terms stand for machine operating, machine relocation, subcontracting costs. The last three terms are similar to those comprising the objective function of MLCLSP model presented Section 6.3. These terms are machine setup, inventory holding and replacement costs of defective parts.

Model Constraints: The constraint in Eq. (6.11) ensures that if a production route of a part is setup, an operation in that route will be assigned to a cell. Eq. (6.12) ensures that setup is performed (i.e. $z_{r,i}(t) = 1$) whenever there is production in period t (i.e. $x_{r,i}(t) > 0$). Eq. (6.13) is inventory balance constraint. It states that the beginning inventories together with the production and subcontracted quantities of each item in each period less the inventory at the end of the period and the defect allowance should meet the external demand and the dependent demand generated

by its non-defective successor items. The constraint in Eq. (6.14) is required to enforce non-negativity on the number of defective items under the JIT-Philosophy. The capacity limitations of the machines are expressed in Eq. (6.15). Eq. (6.16) implies that the number of type k machines used any period is greater than or equal to that of the previous period. This means that the model is not going to remove extra machines of any type if that type of machines happen to be in excess in a certain time period. The presence of extra machines in the system increases system flexibility and reliability by providing alternative routes during machine breakdown. Eq. (6.17) enforces workload balance among cells. Eq. (6.18) states that the number of type k machines in the current period in a particular cell is equal to the number of machines in the previous period, adding the number of machines being moved in and subtracting the number of machines being moved out of the cell. Eq. (6.19) specifies the lower and upper bounds of cell sizes. Eqs. (6.20) and (6.21) set the value of $r_{kl}(t)$ equal to 1 if at least one unit of type k machine is placed in cell l during period t or 0 otherwise. Eq. (6.22) ensures that machine pairs included in Θ are not placed in the same cell. Eq. (6.23) is to ensure that machine pairs included in Ω should be placed in the same cell. Eqs. (6.24) and (6.25) are integrality constraint.

6.4.3. Linearizing the Model

The model presented above is a nonlinear model due to the nonlinear terms in the second term of the objective function, the capacity constraint (Eq. (6.15)) and the workload balancing constraint (Eq. (6.17)). The nonlinear term $x_{ri}(t)\,|\eta_{j+1,ril}(t) - \eta_{jril}(t)|$ in the second term of the objective function can be linearized in two steps. In the first steps, we introduce a binary variable $a_{jril}(t)$ and replace $x_{ri}(t)\,|\eta_{j+1,ril}(t) - \eta_{jril}(t)|$ by the quadratic term $x_{ri}(t) \cdot a_{jril}(t)$ with the additional constraints given

in Eqs. (6.26)–(6.28). In the second step, we replace $x_{ri}(t) \cdot a_{jril}(t)$ by a continuous variable $b_{jril}(t)$ with the additional constraints given in Eqs. (6.29)-(6.31). Similarly, the quadratic terms $x_i(t) \cdot \eta_{jril}(t)$ in the capacity and the workload balancing constraints can be replaced an other continuous variable $c_{jril}(t)$ with the added constraints given in Eqs. (6.32)-(6.34).

$$\eta_{j+1,ril}(t) - \eta_{jril}(t) \leq a_{jril}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{6.26}$$

$$-\eta_{j+1,ril}(t) + \eta_{jril}(t) \leq a_{jril}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{6.27}$$

$$a_{jril}(t) \in \{0,1\} \quad ; \quad \forall(j,r,i,l,t) \tag{6.28}$$

$$b_{jril}(t) \geq x_{ri}(t) + M \cdot a_{jril}(t) - M \quad ; \quad \forall(j,r,i,l,t) \tag{6.29}$$

$$b_{jril}(t) \leq x_{ri}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{6.30}$$

$$b_{jril}(t) \leq M \cdot a_{jril}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{6.31}$$

$$c_{jril}(t) \geq x_{ri}(t) + M \cdot \eta_{jril}(t) - M \quad ; \quad \forall(j,r,i,l,t) \tag{6.32}$$

$$c_{jril}(t) \leq x_{ri}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{6.33}$$

$$c_{jril}(t) \leq M \cdot \eta_{jril}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{6.34}$$

6.5. Numerical Examples

In this section we present a numerical example of the lot sizing model presented in Section 6.3. This numerical example demonstrates the impact of lot size on product quality assuming either the Disruptive- or the JIT-philosophy and is aimed at to show whether these philosophies are correctly model or not. The example has twelve part types to be processed by six machine types in six planning period.

The product structure involving the twelve part types is given in Figure 6.2. Other relevant data for the part and machine types are given in Tables C.1, C.2 and C.3. We assume that there are 4 machines of each type at the beginning of the first planning period. The detail solutions of this numerical example are given in Table C.4 when none of the philosophies are applied, in Table C.5 when JIT-philosophy is applied and in Table C.6 when Disruptive-philosophy is applied. In Table C.4, the shortage related to a given lot size x under JIT-philosophy is calculated as $\alpha \times x - \beta$ while a number of defective items in producing x non-defective parts is calculated as $[(x-\beta)/(1-\alpha)] \times \alpha - \beta$. Similar calculations are done for Disruptive-philosophy and the quantities from each setup are summed up together and given in the last four columns of Table C.4. The summary of the solutions are presented in Figures 6.3, 6.4 and 6.5. In Figure 6.3, it can be seen that for most of the part types there are more setups in JIT-philosophy. This is also reflected in Figure 6.4-(a) by a reduction of average run length from about 1890 to 1500. According to the JIT philosophy, this reduction of run length causes a reduction of average number of defective from about 623 to 520 as shown in Figure 6.4-(b). In Figure 6.3, it can be seen there are lesser number of setups when the Disruptive-philosophy is in place. This reduced number of setups causes the average run length to increase from about 1890 to 3390 as shown in Figure 6.5-(a) and the average number of defects to fall from 828 to 553 as shown in Figure 6.5-(b). The results of the numerical example are in agreement with the philosophy in place.

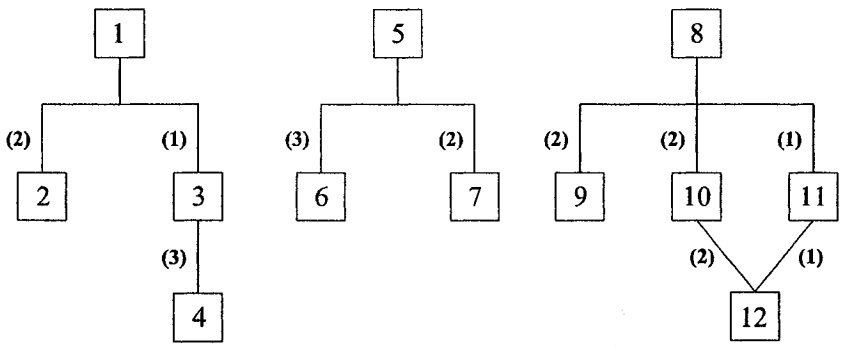


Figure 6.2: Bill-of-Materials for 12 part types

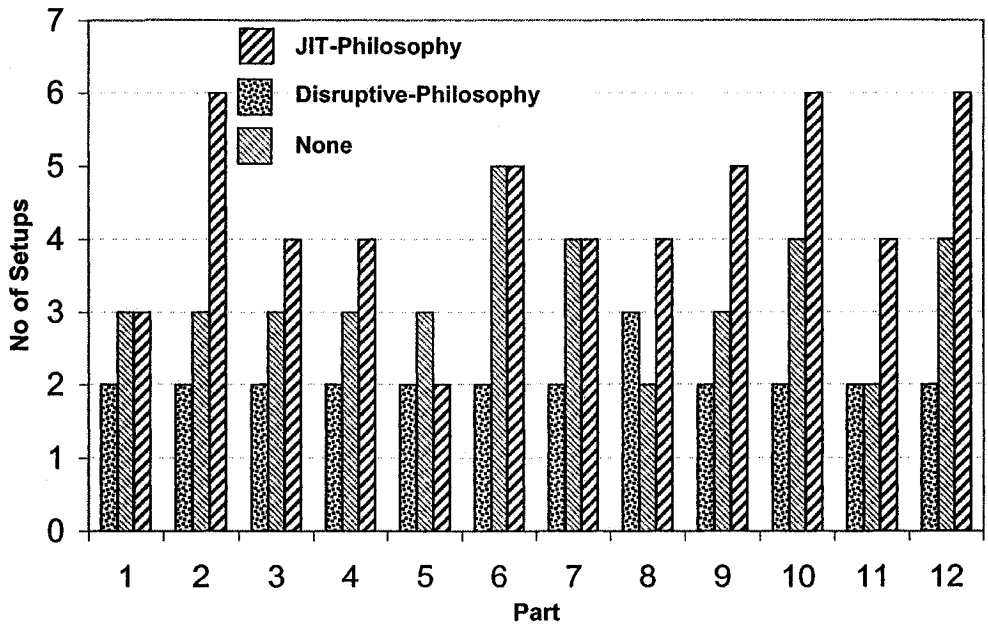


Figure 6.3: No of setups for 12 parts in 6 planning periods

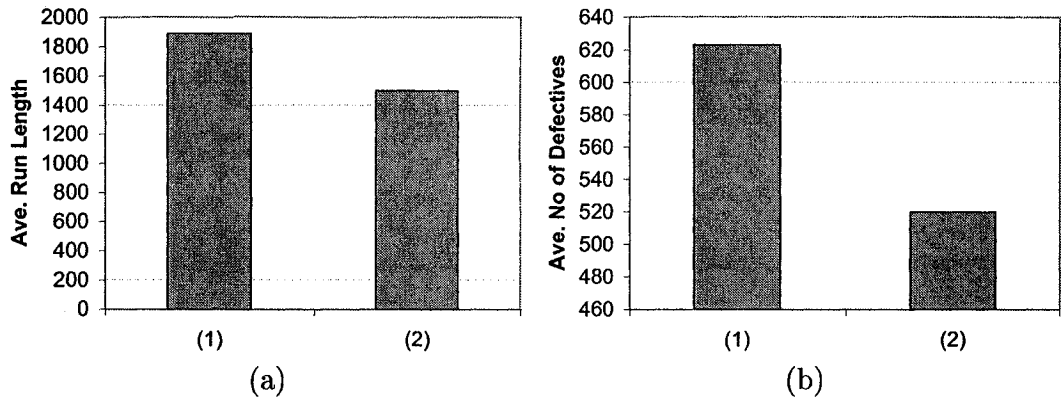


Figure 6.4: (a) Average run length (lot size), (b) average number of defective as per the JIT-philosophy when this philosophy is (1) not applied and (2) applied

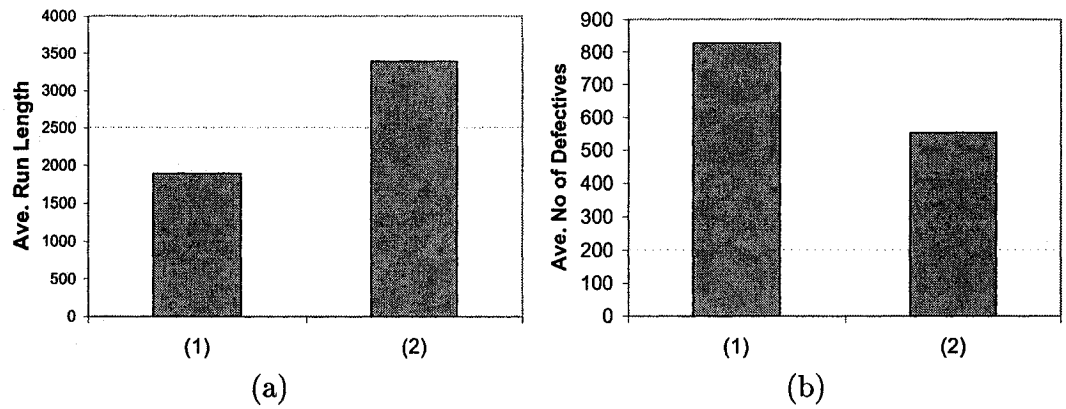


Figure 6.5: (a) Average run length (lot size), (b) average number of defective as per the disruptive philosophy when this philosophy is (1) not applied and (2) applied

Chapter 7

Solution Procedures for Math Model-B

7.1. Introduction

In the previous chapter it was mentioned that an integrated approach should be pursued in manufacturing system analysis, since different aspects of a system are interrelated in many ways. In this endeavor, a comprehensive model consisting of different aspects of the system can help system designers understand the problem better. On the other hand, integrated model may impose computational difficulties and may not be solvable using off-the-shelf optimization software even for small size problems. Thus, efficient heuristic methods are required to solve the proposed model for problems of larger sizes. The heuristic methods developed in Chapters 4 and 5 to solve the model presented in Chapter 3 cannot directly be used to solve the model presented in Chapter 6. One of the basic differences between the mathematical models presented in Chapters 3 and 6 is that in the former case the production quantity in each period is a given data while in the later it is decision variable.

This difference, among others, has brought a significant changes in the solution approaches. In this chapter we present two different heuristics, one based on genetic algorithm and the other based on simulated annealing, to solve the model presented in Chapter 6. During the course of the search, these methods interactively use the simplex algorithm in ILOG CPLEX [60] to solve a Linear Programming (LP) sub-problem corresponding to each solution point visited. Finally, the branch and cut algorithm in ILOG CPLEX is used as a post-optimizer to further improve the solution found by the meta-heuristics. The model of the proposed optimization approach is schematically showing in Figure 7.1 below.

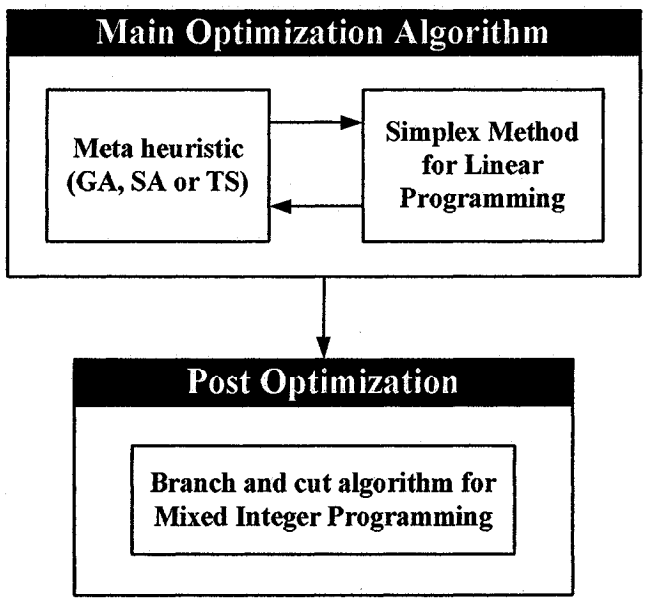


Figure 7.1: The proposed optimization model

7.2. LP Embedded Genetic Algorithm

7.2.1. Chromosomal Encoding of a Solution

As it was stated in Chapter 4, the chromosomal encoding of a solution is the first task in applying a genetic algorithm. In this section, we developed a chromosomal representation of a solution of the model presented in the previous chapter. The solution encoding involves the integer decision variables $N_{kl}(t)$, $\eta_{jril}(t)$, and $z_{ri}(t)$ enabling a randomly generated solution satisfy the constraint in (6.11). The constraints in (6.16), (6.18), (6.20), (6.21), and (6.22) are being taken care by a repair heuristic. Once an integer solution satisfying these constraints is obtained, the corresponding continuous decision variables $x_{ir}(t)$ and $\bar{x}_i(t)$ are determined by solving a Linear Programming sub-problem as explained in the latter section. Figure 7.2 illustrates a chromosome structure assuming 6 part types (P_1 to P_6) are to be precessed in 3 cells (C_1, C_2 and C_3) during T planning period. A segment corresponding to a given time period has two sub-segments: the first sub-segment, labeled 'cells', represents the machine configurations and the second sub-segment, labeled 'parts', represents the operation assignment of the parts to the various cells. In this figure, C_1 in the cells sub-segment and P_1 in the parts sub-segment of period 1 are shown in details. The gene N_{kl} in the chromosome takes a positive integer value. It is the number of machines of type k installed in cell C_l in the period corresponding to its location it the chromosome structure. In the detail for P_1, we assume that part type 1 has two alternative routes labeled $R_{1,1}$ and $R_{2,1}$ with three and two operations respectively. The gene Z_r takes an integer value in {0, 1} to show whether route r is to be setup for production or not. The gene C_{jri} takes the value in $\{1, \cdots, L\}$ and represents the index of the cell in which operation j in route r of part i is to be processed.

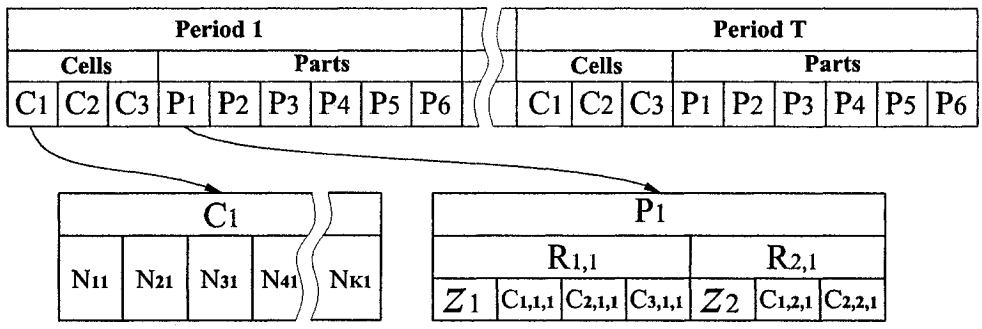


Figure 7.2: Solution representation

7.2.2. Decoding a Chromosome

The values of the variables $N_{kl}(t)$, $z_{ri}(t)$ and $\eta_{jril}(t)$ are determined by decoding a chromosome under consideration. The values of decision variables $N_{kl}(t)$ and $z_{ri}(t)$ are directly picked from chromosome, whereas the decision variable η_{jril} is determined using Eq. (7.1). From this equation, it can be seen that the constraint in Eq. (6.11) can be satisfied by any randomly generated solution as shown in Eq. (7.2).

$$\eta_{jril}(t) = \begin{cases} z_{ri}(t) & ; \text{ If the subscritp } l = \mathrm{C}_{jri} \\ 0 & ; \text{ otherwise.} \end{cases} \tag{7.1}$$

$$\sum_{l=1}^{L} \eta_{jril} = \left(\sum_{\forall l \neq \mathrm{C}_{jri}} \eta_{jril} \right) + \eta_{jri,\mathrm{C}_{jri}} = 0 + z_{ri}(t) = z_{ri}(t) \tag{7.2}$$

Repairing Heuristics

The machine configuration decision $N_{kl}(t)$ directly obtained from the chromosome can be regarded as a preliminary value since the chromosome may violate the constraint in Eq. (6.16) and the machine separation constraint in Eq. (6.22). A chromosome violating the constraint in Eq. (6.16) can be repaired by repairing heuristic similar to the Machine Assignment Heuristic presented in Section 4.2.6. However, in this case $\widetilde{M}_{kl}(t)$, the minimum number of machines of each type required in each cell during each period, is set equal to the preliminary values $N_{kl}(t)$ as obtained from chromosome. Once the value of $\widetilde{M}_{kl}(t)$ is fixed, the steps of the Machine Assignment Heuristic can be followed in order to recalculate $N_{kl}(t)$ so that the constraint in Eq. (6.16) can be satisfied. A chromosome violating the constraint in Eq. (6.22) can be repaired using the repair heuristic presented in Section 4.2.5 with a some variation of the last step of this heuristic. In the solution procedures presented in Section 4, the machine assignment was determined entirely based on the operation assignment. However, in the solution procedure presented in this chapter the machine assignment is directly encoded in the chromosome and it determines the operation assignment. This difference requires a change in the last step of the repair heuristic presented in Section 4.2.5. The last step of the modified repair heuristic is given here under and the other steps need not be changed.

Step 7. For a chromosome under repair, move out all the machine in a given group Υ_p from the cells that are not associated to this group and arbitrarily redistribute these machines among the cells associated to this group.

The last repair to be done on a chromosome is on its operation assignment. If the j^{th} operation in r^{th} route of part i is assigned to a cell in which the required machine type is not available, then the gene C_{jri} has to be reset to the index of one

of the cells containing the machine type m_{jri}. This requires recalculating $\eta_{jril}(t)$ using Eq. (7.1). After the decision variables $N_{kl}(t)$ are determined by decoding and repairing a chromosome, the reconfiguration decision variables $y_{kl}^{+}(t)$ and $y_{kl}^{-}(t)$ can be determined using Eqs. (4.6) and (4.7) respectively. This last step satisfies the constraint in Eq. (6.18) of the proposed model.

LP Subproblem

The values of all the integer decision variables are obtained by decoding a chromosome under consideration and repair heuristics. These integers solution can satisfies all the constraints involving only the integer variables other than the cell size constraint. The corresponding values of the sub-lot sizes $x_{ri}(t)$ and subcontracting quantities $\bar{x}_i(t)$ can be obtained by solving a Linear Programming sub-problem given below. This LP sub-problem is to minimize the sum of inter-cell movement cost, operation cost, inventory holding cost and replacement cost of defective parts subject to the constraints in Eqs. (6.12)–(6.15) and (6.17). In the subproblem, these constraints are renumbered as Eqs. (7.4)–(7.8).

Minimize

$$
\begin{aligned}
Z_{\mathrm{LP}} \;=\; & \frac{1}{2}\sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{i=1}^{P}\sum_{r=1}^{R_i}\sum_{j=1}^{O_{ri}-1} V_i \cdot x_{ri}(t)\,|\eta_{j+1,ril}(t) - \eta_{jril}(t)| \\
& + \sum_{t=1}^{T}\sum_{i=1}^{P}\sum_{r=1}^{R_i}\sum_{j=1}^{O_{ri}} x_{ri}(t) \cdot \lambda_{jri} \cdot O_{m_{jri}} \\
& + \sum_{t=1}^{T}\sum_{i=1}^{P} \Phi_i \cdot \bar{x}_i(t) \\
& + \sum_{t=1}^{T}\sum_{i=1}^{P} H_i \cdot I_i(t) \\
& + \sum_{t=1}^{T}\sum_{i=1}^{P} \Re_i \cdot \sum_{r=1}^{R_i}[\alpha_{ri} \cdot x_{ri} + \beta_{ri} \cdot z_{ri}(t)]
\end{aligned}
\tag{7.3}
$$

Subject to:

$$x_{ri}(t) \leq M \cdot z_{ri}(t) \quad ; \quad \forall(r,i,t) \qquad (7.4)$$

$$I_i(t-1) + \sum_{r=1}^{R_i} x_{ri}(t) + \bar{x}_i(t) - \sum_{r=1}^{R_i}[\alpha_{ri} \cdot x_{ri} + \beta_{ri} \cdot z_{ri}(t)]$$
$$-I_i(t) = d_i(t) + \sum_{j \in \Gamma_i} r_{ij} \cdot \left(\sum_{r=1}^{R_j} x_{rj}(t) + \bar{x}_j(t) - \right.$$
$$\left. \sum_{r}^{R_j}[\alpha_{rj} \cdot x_{rj} + \beta_{ri} \cdot z_{rj}(t)] \right) \quad ; \quad \forall(i,t) \qquad (7.5)$$

$$\alpha_{ri} \cdot x_{ri} + \beta_{ri} \cdot z_{ri}(t) \geq 0 \quad ; \quad \forall(r,i,t) \qquad (7.6)$$

$$\sum_{\forall(j,r,i)|m_{jri}=k} x_i(t) \cdot \eta_{jril}(t) \cdot \lambda_{jii} \leq C_k \cdot N_{kl}(t) \quad ; \quad \forall(k,l,t) \qquad (7.7)$$

$$\sum_{i=1}^{P} \sum_{r=1}^{R_i} \sum_{j=1}^{O_{ri}} x_{ri}(t) \cdot \eta_{jril}(t) \cdot \lambda_{jri} \geq$$
$$\frac{q}{L} \sum_{l=1}^{L} \sum_{i=1}^{P} \sum_{r=1}^{R_i} \sum_{j=1}^{O_{ri}} x_{ri}(t) \cdot \eta_{jril}(t) \cdot \lambda_{jri} \quad ; \quad \forall(l,t) \qquad (7.8)$$

7.2.3. The Fitness Function

As it was stated in Chapter 5, the purpose of the fitness function is to measure the fitness of candidate solutions in the population with respect to the objective and constraint functions of the model. For a given individual, its fitness is given by Eq. (7.9) as the sum of the objective function of the model (Eq. (6.10)) and the penalty terms of constraint violations. The cell size and machine co-location constraint are enforced by such penalty terms. The factors f_{cs} and f_{mc} are used for scaling these penalty terms. Finally, the raw fitness scores F need to be transformed so that the minimum raw fitness will correspond to the maximum transformed fitness. This is achieved using Eq. (4.4).

$$
\begin{aligned}
F &= \text{Model Objective Function} \\
&+ f_{cs} \cdot \sum_{t=1}^{T} \sum_{l=1}^{L} \max \left\{ 0, \quad \sum_{k=1}^{K} N_{kl}(t) - LB_l \ , \quad \sum_{k=1}^{K} N_{kl}(t) - UB_l \right\} \\
&+ f_{mc} \cdot \sum_{t=1}^{T} \sum_{l=1}^{L} \left\{ \sum_{\forall (k^c, k^d) \in \Omega} |r_{k^c l}(t) - r_{k^b l}(t)| \right\}
\end{aligned} \tag{7.9}
$$

7.2.4. Genetic Operators

The genetic operator developed in this chapter are more or less similar to those used in Chapter 4 except these are tailored to the structure of the solution representation shown in Figure 7.2. These operators are explained here under.

Selection Operator: The selection operator used in this chapter is based on the simulation of roulette wheel and it is the same as the one used in Chapter 4.

Crossover Operators: The crossover operators produce children by exchanging information contained in the parents. In this section we present several crossover operators tailored to the structure of the solution representation shown in Figure 7.2. These are:

- Single point crossover,
- Period swamp crossover,
- All-cell swamp crossover,
- All-part swamp crossover,
- Single-cell swamp crossover,
- Single-part swamp crossover and
- Route swamp crossover

Single-point crossover operator is a standard crossover operator used in genetic algorithm. It randomly generates a single crossover point along the length of the chromosome and swamps the right-hand-side segments of the parents. The period-swap crossover operator randomly selects a period in the planning horizon and exchanges the segments of the parent chromosomes corresponding to that period. All-cell (all-part) swap crossover operator randomly selects a cells- (parts-) sub-segment and exchanges this sub-segment of the parents. Single-cell (single-part) swap crossover operator randomly selects a single cell (part) along the length of the chromosome and exchange it between parents. Route swamp crossover randomly selects a production route of a part along the length of the chromosome and exchange the information about this route between parents.

Mutation Operators: The mutation operators act on a single chromosome to alter the information contained in the genes. These operators are usually applied under certain probabilities much less that the crossover probabilities. In this section we present five mutation operators used in the genetic algorithm developed in this chapter. These are:

- Machine mutator,
- Setup mutator,
- Part level cell mutator,
- Route level cell mutator,
- Operation level cell mutator,

Machine mutator applied along the entire length of a chromosome to step up or down the number of machine of each type installed in each cell during each period. The step amount is equal to unity. The setup mutator is applied to each of the $z_{ri}(t)$

in the chromosome to switch its value between 0 and 1. The part-level cell mutator randomly alters the c_{jri}'s of all the operations in all the routes of a part to other identical values in $\{1, 2, \cdots L\}$. The route-level cell mutator randomly alters the c_{jri}'s of all the operations of a given route of a given part to other identical values in $\{1, 2, \cdots L\}$. The part- and route-level mutation operators are applied during the first phase of the genetic search where the quest is to find the best configuration with independent cells. Operation-level cell mutator alters the value of each of c_{jri}'s of the operations of the parts in various route. However, this operator is applied for each operation independently and may result in different values of c_{jri}'s of the operations of a part. Hence, it may result in inter-cell movements. For this reason, this operator is applied in the second phase of the genetic search where the attempt is to optimize the cost of inter-cell movement along other cost terms of the model. Moreover, this operator is applied at a lower mutation rate than part-level cell mutator to avoid unnecessary perturbations.

7.2.5. Two Searching Phases

By following similar reasoning given in section 4.2.8, the LPEGA runs in two phases. In first search phase, single-point crossover and the operation-level cell mutator are not applied as their use can result in solution with inter-cell movement. In the second phase, part-level and route-level cell mutator operators are not applied as these operator only promote the generation of independent cells. The LPEGA also has a procedure called population rejuvenation similar to the one explained in Section 4.2.8.

7.2.6. Steps of the LPEGA

The general sets of the LPEGA are similar to that of the GA presented in Chapter 4. The major difference is that the LPEGA requires solving a linear programming sub-problem corresponding to each individual in the population in each generation. With this difference, the steps of the LPEGA are represented in the flow chart given in Figure 7.3 using the following notations.

PS	Population size
ρ	Index for individual in a given population
g	Generation counter, $\rho = 1, 2, ..., \rho_{max}$,
g_{max}	Maximum number of generations,
$Phase$	The current phase of the search which equals to 1 for the first phase or 2 for the second phase,
g_{phase}	Generation at which the value of $Phase$ should be set equal to 2 if it were not previously set to this value by other conditions,
w	Number of successive generations counted without any improvement of the best individual so far found,
w_{max}	Maximum value of w at which point population rejuvenation is to be performed,
b	Number of successive population rejuvenations counted without any improvement of the best individual so far found,
b_{max1}	Maximum value of b at which point the second phase is to be entered if $Phase$ was equal to 1,
b_{max2}	Maximum value of b in the second phase at which point the search will be terminated.

f An indicator binary variable which equal to 1 if population rejuvenation is to be accomplished; 0 otherwise.

The steps were coded in C++ and run on a Pentium-4 (2.93 GH, 512 MB Ram) PC computer. Simplex subroutines within ILOG CPLEX package [60] was used to solve the embedded linear programming model.

7.3. LP Embedded Simulated Annealing

In this section we developed simulated annealing algorithm to solve the model presented in the Chapter 6. Similar to the genetic algorithm presented above, the proposed simulated annealing uses the simplex method to solve a linear programming sub-problem corresponding to each solution visited. The simulated annealing is based on a SMMC-SA presented in Section 5.2.2. The components and steps of this LP embedded simulated annealing (LPESA) are explained in the next sections.

7.3.1. Components of LPESA

LPESA shares many similar components and features of LPEGA. Solution representation and decoding, repair heuristics, perturbation operators (mutation operators), energy function (fitness function) and two search phases are common to both LPESA and LPEGA. The cooling schedule used in LPESA is same as the one presented in Section 5.2.4. The Markov chains in LPESA interact in similar fashions as that of the parallel simulated annealing (Section 5.2.4) except that the interactions in the LPESA are limited within a single computer.

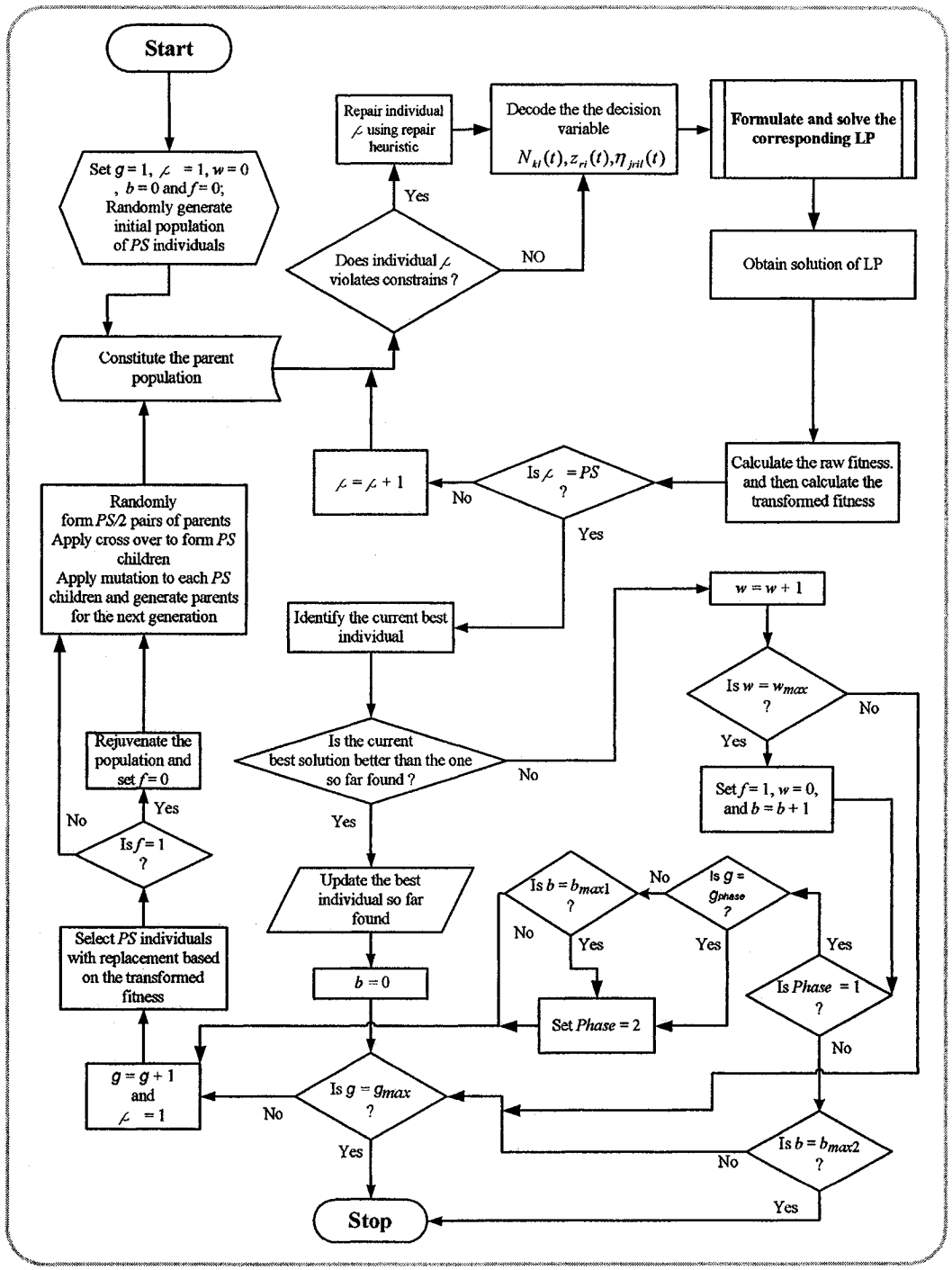


Figure 7.3: The steps of LPEGA

7.3.2. Steps of the LPESA

The steps of LPESA based on a periodic interaction scheme of the Markov chains are presented in Figure 7.4. The notations used in presenting the algorithm are given below.

s	Index of Markov chain, $s = 1,\ 2,\ ...,\ S$ where S is the number of Markov chains followed
n	Iteration counter, $n = 1,\ 2,\ ...,\ N$ where N is the maximum number of iterations in each Markov chain
X_{ns}	The solution at the n^{th} iteration along the s^{th} Markov chain
α	Cooling schedule exponent
r	Index for the temperature levels in the cooling schedule
T_r	Temperature at the r^{th} level, $T_r = \alpha \times T_{r-1} = \alpha^r \times T_0$
Q	Number of iterations to be performed in each Markov chain at each temperature level
F	Interaction frequency. It is defined as the number of iterations to be performed by each Markov chain before interaction is effected among these Markov chains
BI	Best feasible individual so far found
$Phase$	The current phase of the search which equals to 1 for the first phase or 2 for the second phase
C_{phase}	The number of iterations to be performed by each Markov chain before the search phase is changed from $Phase = 1$ to $Phase = 2$

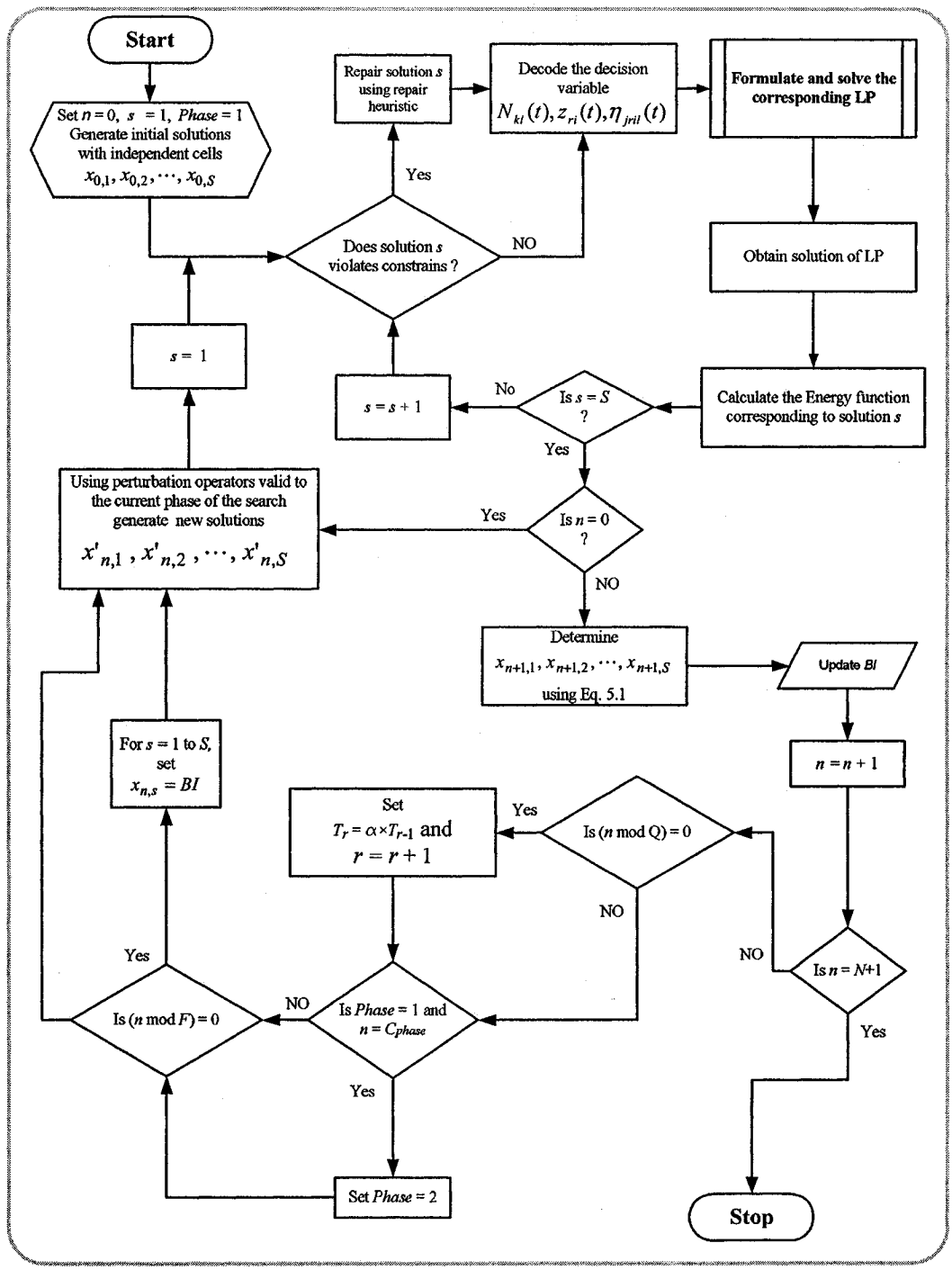


Figure 7.4: The steps of LPESA

7.4. Branch and Cut Algorithm as Post-optimizer

Mungwattana [107] recommended, as a future research, the use of CPLEX as a post optimizer in solving cell formation models where good initial solutions can be generated using heuristic methods. Following his recommendation, we use the branch and cut algorithm in ILOG CPLEX as a post-optimizer. The solution generated by the GA (or SA) is used as the first incumbent solution for the branch and cut algorithm.

7.5. Computational Performances

The problem presented in Section 6.5 was used to evaluate the performance of the proposed heuristic in generating two manufacturing cells. In Figure 7.5 is the convergence of the proposed GA followed by the branch and cut algorithm as a post optimization. In this figure the best solution so far know after several hours using CPLEX and two lower bounds on the solution are also shown. The lower bound determined by the CPLEX is very low compared to the solution determined by the proposed heuristic. The second and improved lower bound was determined by solving the model to optimality after relaxing the machine separation constraint and allowing all the machine to be installed in a single big cell. Thus the optimal solution of the original model is well above this lower bound. This implies that the proposed heuristic has found a solution very close to global optimal solution. Similar computational performance was also observed using the simulated annealing based heuristic.

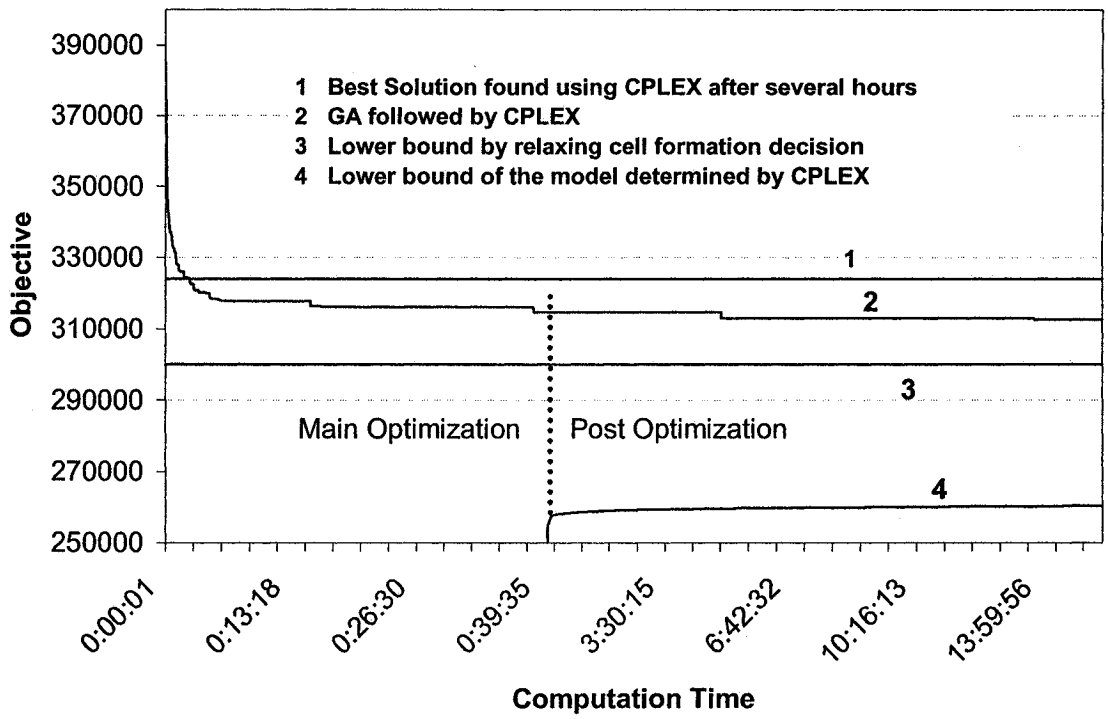


Figure 7.5: The convergence of the proposed heuristic for the example problem

Chapter 8

Summary, Conclusion and Future Research

8.1. Summary and Conclusion

In the past few decades, there has been an increasing worldwide awareness towards productivity improvement. A new style of operation and a new environment in the work place conductive to improvement in such factors as flexibility, efficiency, management-worker relation, team work and job satisfaction are becoming important for survival in the international market. CMS has emerged as one of the promising strategies to address such requirements. The contributions of this research lie in developing (1) comprehensive mathematical models for CMS design problems and (2) efficient solution procedures to solve the model developed.

8.1.1. Contributions in Modeling CMS Problems

The design of CMS involves many structural and operational issues. One of the first important design steps is the formation of part families and machine cells.

The effectiveness of this design step heavily depends on the proper consideration of relevant issues. From our review of recently published cell formation methods we noticed that these factors include:

1. Alternative routing
2. Demand fluctuation
3. Dynamic cell reconfiguration
4. Workload balancing
5. Lot-splitting
6. Types of tools required by a part
7. Types of tools available on a machine
8. Machine proximity constraint
9. Sequence of operation
10. Setup cost/time
11. Cell / part family size constraint
12. Movement of parts (material handling cost)
13. Machine capacity
14. Presence of identical machines
15. Machine investment cost
16. Subcontracting cost
17. Tool consumption cost
18. Unit operation time
19. Operation cost
20. Product structure (bill of materials)
21. Production planning

However, our review also showed that individual methods published recently addressed only a limited subsets of these factors. To this end, a model that incorporates all or the majority of these factors is essential. In Chapter 3, a comprehensive mathematical model for the design of cellular manufacturing systems is proposed incorporating more of these attributes than recently published articles. The model incorporates the attributes from 1 through 19 in the list above. Numerical examples using small size problems, all solved by LINGO, were presented to demonstrate the model and its potential benefits. Computational experience on such small problems showed that a significant amount of cost savings can be achieved by considering system reconfigurations, lot splitting and system flexibility. Our computational results

also showed that there are significant differences on workload distribution among the cells, if workload balancing is not attempted. The result is compiled in a paper and accepted for publication in ***International Journal of Production Economics***.

In the model developed in Chapter 3, the production quantity of a part in a given time period is a given data and equals to the demand of the part in that time period. However, a production quantity can be a decision variable to be determined by the production planning function. Principally, production quantity (lot size) depends on the setup cost, inventory holding cost and capacity constraints. Production capacity in turn depends, among others, on the type and number of machines installed in the manufacturing cells which is a cell formation decision. Thus, production planing and dynamic cell formation can be viewed as interdependent problems which need an integrated approach. Based on this considerations, in Chapter 6, we expand the model presented in Chapter 3 to address dynamic cell formation and a multi-item multi-level capacitated lot sizing problems in an integrated manner. The model also considers the effect of lot sizes on the product quality. Proponents of the Just-In-Time philosophy contend that smaller lot sizes result in improved product quality. Others (from the disruptive philosophy) argue that smaller lot sizes result in excessive interruptions and disruptive environments that impair learning and increase defects. The intent in our model is not to justify or condone one philosophy over the other; i.e. whether short production runs have a positive effect (JIT philosophy) or a negative effect (disruptive philosophy) on product quality. Rather, it is our intention to incorporate this frequently identified influence of run length on product quality into an integrated dynamic cell formation and production control model that can be utilized under either circumstance.

8.1.2. Contributions in Developing Solution Methods

In the previous section, we mention that one of the major contribution of this research is in developing a comprehensive mathematical programming model for dynamic cellular manufacturing system design. In solving such a comprehensive mathematical programming model for large size problems, branch and bound general search algorithms cannot give optimal or near optimal solutions within acceptable computational times. To this end, in Chapter 4, we developed an efficient heuristic based on genetic algorithm to solve the comprehensive model presented in Chapter 3. The algorithm incorporates several problem specific genetic operators, constraint handling techniques and divided searching phases. Its performance was evaluated against available off-the-shelf optimization software. The results obtained by using the heuristic method were very encouraging and a paper reporting these results has been accepted for publication in the ***International Journal of Production Research***. In order to further improve the obtained results, a parallel implementation of the algorithm (PGA) is also attempted. Though researchers agreed that CMS design is a complex problem and tried to develop efficient solution procedures for the last decades, there have been a few report on the use of PGA for CMS design. As yet, the PGA methodology has not been exploited for the design of CMSs and so this novel approach is attempted in this research. The parallel implementation of the algorithm demonstrates a reduction of processing time and improved search performance. Thus, with this work, we could ratify the importance of using parallel genetic algorithm in CMS design where there are a few reports on its use.

In Chapter 5, we developed an alternative solution approach based on simulated annealing to solve the model presented in Chapter 3. Most SA schemes in literature follow a single Markov chain. However, following a single Markov chain

may not be necessary from a performance point of view. Hence, we developed a multiple Markov chain simulated annealing algorithm which allows multiple search directions to be traced simultaneously. Our computational results showed that the multiple Markov chain SA is more efficient than the conventional single Markov chain SA. To further improve the performance of the proposed multiple Markov chain SA, we implement it in a parallel computing environment. The parallelization was achieved by distributing several Markov chains to concurrently available processors. The parallel implementation greatly improved computational efficiency of the multiple Markov chain SA in terms of solution quality and computational time. The result was compiled in paper and submitted to ***Computers & Industrial Engineering*** for possible publication.

The heuristic methods developed to solve the model presented in Chapter 3 cannot directly be used to solve the model presented in Chapter 6. As it was pointed out in the previous section, one of the basic differences between the mathematical models presented in Chapters 3 and 6 is that in the former case the production quantity in each period is a given data while in the later it is decision variable. This difference, among others, has brought a significant changes in the solution approaches. In Chapter 7 we present GA and SA based different heuristics in order to efficiently solve the model presented in Chapter 6. In these search heuristics, only the integer variables were directly encoded in a solution representation. The structure of the solution representation and repair heuristics enable a randomly generated integer solution to satisfy several constraints composed of the integer variables. For a given integer solution, the corresponding continuous variables were determined by solving a linear programming model minimizing several cost terms and satisfying constraints with continuous variables. This strategy leads to a highly reduced search space resulting in a lesser number of solution points that may be

potentially visited by the search methods to reach optimal or near optimal solution. The simplex algorithm in ILOG CPLEX was used to solve a Linear Programming (LP) sub-problem corresponding to each solution point visited. The performances of the heuristics were very encouraging in solving the integrated cell formation and production planning model. Finally, The branch and cut algorithm is used as a post-optimizer to further improve the solution found by the meta-heuristics.

8.2. Future Research

8.2.1. Second Phase of CMS Design

As it has been stated in Section 1.3, the second phase of CM design consists of system design of each of the individual cells. Typical decisions in this phase include:

1. Equipment layout
2. Design/selection of material handling equipment
3. Assessment of operators training requirement
4. Operators assignment
5. Machine loading and scheduling
6. Job dispatching
7. Maintenance planning
8. Quality control and inspection policies

However, we have felt that the consideration of the second phase of CM design is beyond the scope of this thesis. Its consideration will require substantial time and cannot be fitted within the time frame of the work of this thesis. For this reason, the second phase of CM design will be considered in our future research work.

8.2.2. Multi-objective Optimization

The heuristic methods developed in this research can give only one optimal or near optimal solution for the proposed mathematical models. These heuristic methods will be modified and improved to generate several near optimal alternative solutions. Then, a simulation model will be developed and used to further evaluated the alternative solutions for their machine utilization, work-in-process, due date performance, reliability, flexibility and other performance measures that are difficult to express in monetary terms.

Appendix A

Data and Solutions for Chapter 3

Table A.1:Parts'data

Part No.	Batch size B_i	Cost of inter-cellular movement per unit V_i	Demand during period t	
			$t=1$	$t=2$
1	100	6	4000	0
2	150	12	0	3200
3	300	27	4000	2500
4	200	18	0	4500
5	100	15	4400	0
6	100	15	0	4500
7	150	24	0	4500
8	150	12	3600	0
9	150	12	3400	0
10	200	18	6500	0
11	300	24	0	4500
12	100	12	0	3550

Table A.1: Continued

Part No.	Batch size B_i	Cost of inter-cellular movement per unit V_i	Demand during period t	
			$t = 1$	$t = 2$
13	100	27	2000	6000
14	150	21	4000	0
15	300	27	4400	4500
16	150	24	0	6000
17	200	12	3500	0
18	100	21	3800	5700
19	100	18	4800	0
20	100	12	0	3800
21	100	24	0	3000
22	150	15	4700	3000
23	120	18	5400	0
24	150	27	0	4500
25	150	18	0	4500

Table A.2: Tool requirement of parts (tool codes assumed)

Part No.	Tool required for operation j (Sequence of Operations)								
	1	2	3	4	5	6	7	8	9
1	A01	C01	A03	C04	C06				
2	G01	G02	G06	F01	F03	F02	H02	H01	H03
3	B01	B02	B03	A02	A01	E01	E03	E02	
4	D01	D02	D06	H01	H03				
5	G05	H02	H03	F02	F04	F05	F03	G03	G04
6	D02	D04	D06	A01	A02	H02			
7	A01	A02	A03	C02	C03	C06	A04	A05	D05
8	B01	B02	E01	E02	B03	B04			
9	FO4	F05	G04	H03	H04	G05	G06		
10	D01	D02	D03	D05	H04	H05			
11	A01	C01	C03	A03	B01	E03			
12	B03	B04	A01	F01	E03				
13	F03	F02	F01	G01	G02	G04	D04	D03	D05
14	E01	E02	E03	D02	D06				
15	A03	C02	C03	A05	C06	G06			
16	G05	G06	F01	F02	F05				
17	A02	B03	B04	E03					
18	C01	C02	C04	A01	C06				
19	D03	D04	D05	H01	H03	H02			
20	B01	B02	B03	A01	A02	B04	E02	E03	
21	H04	H05	F03	F04	F05	E01	G01	G02	
22	A01	A03	C01	C03	B01	E03			
23	A01	A02	D02	D04	D06	H01			
24	E01	E02	B02	B01	B04				
25	F04	F05	G03	G05	G06	H04			

Table A.3: Tool availability on machines

Machine type k	Available tools
1	A01, A02, A03, A04, A05
2	A01, A02, B01, B02, B03, B04
3	C01, C02, C03, C04
4	C03, C04, C05, C06
5	D01, D02, D03, D04, D05, D06
6	E01, E02, E03
7	F01, F02, F03, F04, F05
8	G01, G02, G03, G04, G05, G06
9	H01, H02, H03, H04, H05
10	H01, H02, H03

The tool index $g = 1$ for AO1, 2 for A02, ..., and 40 for H05.

Table A.4: Routes and alternative routings data (k, μ_{ijk}, h_{ijk})

Part No.	Operation No. 1	2	3	4	5	6	7	8	9
1	(1,100,12) (2,110,11)	(3,80,18)	(1,150,10)	(3,120,10) (4,155,8)	(4,90,16)				
2	(8,40,16)	(8,40,14)	(8,40,16)	(7,90,12)	(7,120,12)	(7,90,10)	(9,100,10) (10,120,10)	(9,130,17) (10,120,18)	(9,180,16) (10,120,18)
3	(2,90,5	(2,90,10)	(2,90,15)	(1,100,10) (2,120,9)	(1,100,16) (2,120,15)	(6,30,20)	(6,40,15)	(6,40,10)	
4	(5,30,8)	(5,30,10)	(5,90,12)	(9,70,16) (10,140 15)	(9,70,20) (10,120,19)				
5	(8,60,9)	(9,90,10) (10,120,9)	(9,60,12) (10,120,11)	(7,90,6)	(7,120,8)	(7,100,8)	(7,120,10)	(8,40,16)	(8,60,12)
6	(5,30,16)	(5,60,10)	(5,90,8)	(1,100,12) (2,100,12)	(1,100,16) (1,120,14)	(9,100,12) (10,120,12)			
7	(1,100,5) (2,120,5)	(1,100,8) (2,120,9)	(1,150,6)	(3,80,10)	(3,120,12) (4,140,11)	(4,90,14)	(1,150,16)	(1,150,10)	(5,70,8)
8	(2,90,6)	(2,90,8)	(6,30,12)	(6,40,14)	(2,110,8)	(2,110,8)			
9	(7,120,6)	(7,100,12)	(8,60,16)	(9,60,14) (10,120,13)	(9,80,20)	(8,60,6)	(8,60,16)		
10	(5,30,12)	(5,30,6)	(5,60,8)	(5,70,16)	(9,80,7)	(9,80,4)			
11	(1,100,7) (2,90,7)	(3,80,10)	(3,120,10) (4,140,9)	(1,150,15)	(2,90,8)	(6,40,5)			
12	(2,110,7)	(2,110,11)	(1,100,8) (2,120,7)	(7,90,15)	(6,40,15)				
13	(7,20,5)	(7,90,14)	(7,90,8)	(8,40,10)	(8,40,6)	(8,60,5)	(5,60,10)	(5,60,12)	(5,70,12)
14	(6,30,16)	(6,40,8)	(6,40,12)	(5,30,6)	(5,90,4)				
15	(1,150,5)	(3,80,10)	(3,120,12) (4,140,11)	(1,150,5)	(4,90,8)	(8,60,4)			
16	(8,60,12)	(8,60,10)	(7,90,14)	(7,90,9)	(7,100,14)				
17	(1,100,16) (2,120,16)	(2,110,6)	(2,110,18)	(6,40,12)					
18	(3,80,5)	(3,80,8)	(3,130,10) (4,140,9)	(1,100,8) (2,120,7)	(4,90,16)				
19	(5,60,10)	(5,60,9)	(5,70,13)	(9,70,12) (10,120,11)	(9,60,12) (10,110,11)	(9,70,16) (10,120,15)			
20	(2,90,8)	(2,90,6)	(2,110,6)	(1,100,9) (2,100,9)	(1,100,16) (2,110,15)	(2,110,7)	(6,40,15)	(6,40,12)	
21	(9,80,14)	(9,80,10)	(7,120,8)	(7,120,5)	(7,100,5)	(6,30,12)	(8,40,18)	(8,40,12)	
22	(1,100,18) (2,105,17)	(1,150,8)	(3,80,14)	(3,140,16) (4,125,15)	(2,90,12)	(6,40,12)			
23	(1,100,8) (2,120,7)	(1,110,20) (2,140,18)	(5,30,7)	(5,60,9)	(5,90,12)	(9,70,12) (10,125,11)			
24	(6,30,14)	(6,40,16)	(2,90,8)	(2,90,10)	(2,110,8)				
25	(7,120,8)	(7,120,6)	(8,40,12)	(8,60,10	(8,60,6)	(9,80,5)			

Table A.5
Machines' data

Machine type k	G_k	O_k	C_k	R_k^+	R_k^-	$Q_k(t)$		$P_k(t)$	
						$t=1$	$t=2$	$t=1$	$t=2$
1	500.00	12.00	2000	75.00	75.00	7	1	12500.00	12700.00
2	600.00	11.00	1800	100.00	100.00	12	1	11800.00	12200.00
3	800.00	8.00	1800	140.00	140.00	10	3	10000.00	10200.00
4	400.00	10.50	2000	90.00	90.00	7	2	11200.00	11200.00
5	900.00	8.50	2000	80.00	80.00	8	4	17200.00	17200.00
6	450.00	10.00	2200	100.00	100.00	9	1	13200.00	13200.00
7	650.00	9.00	1840	70.00	70.00	10	2	16200.00	16200.00
8	450.00	8.00	1800	70.00	70.00	10	1	15000.00	15000.00
9	650.00	13.00	2200	80.00	80.00	10	2	12200.00	12200.00
10	300.00	12.00	2200	75.00	75.00	10	4	13200.00	13200.00

Table A.6: Miscellaneous data

No of cells	3
Lower bound for the cell size	2 machines
Upper bound for the cell size	25 machines
Pair of machines that should not be located in the same cell (arbitrarily selected)	{2, 4} and {6, 9}
Work load balancing factor, q	0.90

Table A.7: Part-cell assignment for period 1

Cell	Machine Type	Machine Qnt.	Parts Types 1	5	9*	10*	23	9*	10*	13*	15	18	19	3	8	13*	14	17	22
	M1	2	1				1												
	M3	1	1																
	M4	1	1																
C1	M5	2				0.33	1												
	M7	2		1	0.41														
	M8	2		1	0.41														
	M9	1		1	0.41	0.33	1												
	M10	1		1	0.41		1												
	M1	1									1	1							
	M3	1									1	1							
	M4	2									1	1							
C1	M5	3							0.67	0.31			1						
	M7	2						0.59		0.31									
	M8	1						0.59		0.31	1								
	M9	1						0.59	0.67				1						
	M10	2						0.59					1						
	M1	1																	1
	M2	6												1	1			1	1
	M3	2																	1
C3	M5	1														0.69	1		
	M6	4												1	1		1	1	1
	M7	1														0.69			
	M8	2														0.69			

The numbers in the body of the table indicate the proportion of the total demand of parts whose operations are performed in the corresponding cell.

* Parts appearing in more than one column of this table represent lot splitting

Table A.8: Part-cell assignment for period 2

Cell	Machine Type	Machine Qnt.	Parts Types																
			3	11	12	16*	20	21	6*	7	13*	15	16*	18	2	4	6*	13*	25
C1	M1	1		1			0.94												
	M2	6	1	1	1		0.94												
	M3	2		1															
	M6	4	1		1		1	1											
	M7	1			1	0.13		1											
	M8	1				0.13		1											
C2	M1	3							0.18	1		1		1					
	M3	2								1		1		1					
	M4	3								1		1		1					
	M5	2							0.18	1	0.66								
	M7	3						0.60			0.66		0.87						
	M8	2									0.66	1	0.87						
	M9	1						0.60	0.18										
C3	M2	2					0.06										0.82		
	M5	1														1	0.82	0.34	
	M7	3						0.40							1			0.34	1
	M8	3						0.40							1			0.34	1
	M9	3													1	1	0.82		1
	M10	1													1	1			

Numbers outside the diagonal blocks indicate the presence of intercell movement.

Table A.9: Sample values of the decision variable $\eta_{jikl}(t)$

Period	Part No.	Cell	Machine, $\eta_{jikl}(t)$ Operation 1	2	3	4	5	6	7	8
1	15	1	1, 1.00	3, 1.00	4, 1.00	1, 1.00	4, 1.00	8, 1.00		
1	1	1	1, 1.00	3, 1.00	1, 1.00	3, 0.90 4, 0.10	4, 1.00			
1	10	1	5, 0.33	5, 0.33	5, 0.33	5, 0.33	9, 0.33	9, 0.33		
		2	5, 0.67	5, 0.67	5, 0.67	5, 0.67	9, 0.67	9, 0.67		
2	6	2	5, 0.18	5, 0.18	5, 0.18	1, 0.18	1, 0.18	9, 0.18		
		3	5, 0.82	5, 0.82	5, 0.82	2, 0.82	2, 0.82	9, 0.82		
2	21	2	9, 0.60	9, 0.60	7, 0.60	7, 060				
		3	9, 0.40	9, 0.40	7, 0.40	7, 040	7, 0.40			
		1					7, 0.60	6, 1.00	8, 1.00	8, 1.00

Table A.10
Generic attributes of the 10 additional problems

Problem No.	No of Planning Periods	No of Cells	No of Machines Types	No of Part Types	No of potentially non zero variables		No of Constraints
					Integer	Total	
2	2	3	10	25	2120	6392	5694
3	2	3	10	25	2024	6098	5676
4	2	3	6	25	2040	6190	5376
5	2	3	6	25	1932	5860	5420
6	2	3	6	15	1392	4186	3664
7	2	3	6	15	1392	4186	3772
8	3	3	6	15	2088	6275	5506
9	3	3	6	15	2088	6284	5579
10	2	4	8	20	1856	5474	4928
11	2	4	8	20	5474	1856	4993

Appendix B

GA Parameters for Chapter 4

Table B.1: Values of the parameters of GA used for 13 test runs of Example 1

	Test Run												
Parameter Name	1	2	3	4	5	6	7	8	9	10	11	12	13
Population Size	360	360	540	390	420	426	570	285	430	340	380	556	340
Crossover probability for													
Period-swamp crossover	0.6	0.6	0.9	0.85	0.55	0.45	0.9	0.74	0.76	0.65	0.77	0.72	0.65
Part-swamp crossover	0.6	0.6	0.8	0.75	0.85	0.82	0.9	0.82	0.9	0.55	0.66	0.76	0.75
Single point crossover	0.4	0.4	0.5	0.25	0.35	0.4	0.35	0.52	0.56	0.4	0.35	0.35	0.4
Operation-swamp crossover	0.1	0.1	0.3	0.15	0.2	0.3	0.4	0.32	0.26	0.1	0.18	0.18	0.1
Mutation Probability for													
Alternative route mutator	0.05	0.04	0.03	0.02	0.003	0.006	0.015	0.012	0.016	0.035	0.035	0.035	0.035
Subcontract mutator	0.004	0.006	0.02	0.012	0.03	0.02	0.015	0.014	0.018	0.01	0.01	0.02	0.01
Part-level cell mutator	0.004	0.006	0.008	0.01	0.04	0.007	0.012	0.015	0.007	0.006	0.006	0.005	0.005
Operation-level cell mutator	0.0004	0.0006	0.002	0.001	0.002	0.0006	0.003	0.003	0.0005	0.0006	0.0006	0.0005	0.0006
Degeneration limit													
d_1	0.06	0.06	0.06	0.04	0.05	0.05	0.05	0.07	0.07	0.05	0.05	0.05	0.05
d_2	0.06	0.06	0.06	0.04	0.04	0.04	0.04	0.07	0.07	0.05	0.05	0.05	0.05
Step amount for													
Subcontract mutator	0.15	0.2	0.05	0.07	0.07	0.06	0.05	0.065	0.072	0.2	0.06	0.06	0.15
Alternative route mutator	0.1	0.06	0.05	0.1	0.1	0.08	0.05	0.065	0.082	0.06	0.06	0.06	0.06
φ_{max}	15	15	10	10	8	12	12	8	10	12	14	12	12
ϖ_{max1}	100	100	120	120	200	220	220	195	212	100	100	100	100
Penalty cost factor for													
Workload balancing , f_{wb}	0.001	0.001	0.0005	0.006	0.002	0.005	0.005	0.015	0.008	0.001	0.001	0.001	0.001
Cell size , f_{cs}	1000	1000	800	850	900	1000	1000	1150	1000	1000	1020	1020	1000

Appendix C

Data and Solution for the Example Problems in Chapter 6

Table C.1
Data for the parts for Section 6.5

	Costs related to				Independent Demand in Period t					
Part	Inventory holding	Replacing Defective	Inter-cell Movement	Sub-contracting	1	2	3	4	5	6
1	1.4	5	1.2	16	500	500	400	0	300	500
2	1.4	10	1.8	24	100	400	200	300	0	400
3	1.4	10	1.8	24	230	0	350	400	320	0
4	0.7	5	1.6	16	200	250	400	350	200	450
5	1.4	10	1.5	24	100	800	600	400	0	600
6	2.1	10	2.4	24	100	500	200	600	250	300
7	1.4	10	1.2	24	0	400	600	300	500	0
8	1.4	5	1.2	24	200	300	600	400	500	100
9	2.1	10	1.1	24	100	200	0	200	0	400
10	2.1	5	2.1	28	0	200	500	0	200	300
11	1.4	10	1.2	28	200	500	0	520	600	300
12	1.4	10	2.4	28	100	0	400	300	0	600

Table C.2
Data for the parts for Section 6.5 (Continued)

			Defect Rate		Operation Sequence(m_{jri}, λ_{jri})		
			JIT	Disruptive	Operation		
Part	Route	Setup	(α, β)	(α, β)	1	2	3
1	1	920	(0.10, 48)	(0.02, 180)	(3, 3)	(6, 4)	
2	1	900	(0.15, 56)	(0.03, 210)	(1, 4)	(3, 5)	(6, 5)
	2	920	(0.20, 56)	(0.04, 210)	(1, 4)	(4, 5)	(3, 5)
3	1	400	(0.15, 40)	(0.03, 150)	(2, 2)	(5, 3)	(4, 3)
4	1	920	(0.10, 40)	(0.02, 150)	(3, 3)	(5, 4)	(2, 3)
5	1	930	(0.05, 48)	(0.01, 180)	(2, 4)	(6, 4)	
6	1	770	(0.10, 48)	(0.02, 180)	(2, 3)	(3, 3)	(6, 2)
	2	700	(0.10, 56)	(0.02, 210)	(2, 4)	(6, 4)	
	3	800	(0.15, 56)	(0.03, 210)	(2, 2)	(1, 3)	(4, 3)
7	1	800	(0.10, 56)	(0.02, 210)	(6, 4)	(2, 5)	
	2	820	(0.10, 64)	(0.02, 240)	(2, 3)	(5, 3)	(3, 3)
8	1	890	(0.10, 48)	(0.02, 180)	(1, 4)	(4, 4)	(2, 4)
9	1	900	(0.15, 64)	(0.03, 240)	(6, 3)	(3, 3)	
10	1	930	(0.15, 48)	(0.03, 180)	(1, 4)	(4, 3)	(3, 4)
	2	780	(0.15, 64)	(0.03, 240)	(3, 5)	(1, 4)	(4, 2)
11	1	920	(0.10, 64)	(0.02, 240)	(1, 3)	(4, 4)	(3, 3)
	2	940	(0.15, 64)	(0.03, 240)	(5, 3)	(4, 6)	
	3	990	(0.10, 56)	(0.02, 210)	(1, 2)	(4, 3)	(5, 4)
12	1	890	(0.10, 64)	(0.02, 240)	(2, 2)	(6, 3)	

Table C.3
Data for the machines for Section 6.5

Machine Type	Procurement Cost	Holding Cost	Capacity (hrs)	Installation Cost	Uninstalling cost	Operation Cors/hrs
1	1120	200	340	60	60	8
2	1000	100	320	80	80	9
3	1250	150	300	80	80	7
4	1720	200	310	90	90	7
5	1320	100	340	60	60	8
6	1400	200	350	70	70	7

Table C.4
Lot sizes when neither JIT- nor disruptive-philosophy is applied

	Selected Route Lot Size Inventory Level in period t =						No of Setups Ave. Lot Size	Total Shortage		No of Defective	
Part	1	2	3	4	5	6	Ave. Inventory	JIT	Disr.	JIT	Disr.
1	1	-	1	-	1	-	3				
	1000	-	400	-	800	-	733	84	584	93	596
	500	0	0	0	500	0	167				
2	1	-	1	-	1	-	3				
	2500	-	1300	-	1999	-	1933	702	804	826	829
	400	0	300	0	400	0	183				
3	1	-	1	-	1	-	3				
	1230	-	1150	-	1120	-	1167	405	555	476	572
	0	0	400	0	0	0	67				
4	1	-	1	-	1	-	3				
	4140	-	4200	-	4010	-	4117	1115	697	1239	711
	250	0	350	0	450	0	175				
5	1	-	1	-	-	1	3				
	899	-	999	-	-	599	833	2	565	2	571
	800	0	400	0	0	0	200				
6	1	2	1	2	-	2	5				
	2798	500	3198	850	-	2099	1890	687	1179	763	1203
	0	0	0	250	0	0	42				
7	2	-	2	-	1	1	4				
	2199	-	2899	-	500	1199	1700	446	1036	496	1057
	400	0	300	0	0	0	117				
8	1	-	-	1	-	-	2				
	1100	-	-	1000	-	-	1050	114	402	127	410
	900	600	0	600	100	0	367				
9	1	-	-	1	-	1	3				
	2499	-	-	2200	-	400	1700	577	873	679	900
	200	0	0	0	0	0	33				
10	2	-	2	2	-	2	4				
	2400	-	499	2200	-	300	1350	573	1122	674	1157
	200	0	0	200	0	0	67				
11	2	-	-	2	-	-	2				
	1799	-	-	2419	-	-	2110	505	607	594	625
	500	0	0	900	300	0	283				
12	1	-	1	1	-	1	4				
	6700	-	1399	7120	-	1200	4105	1386	1288	1540	1315
	0	0	0	0	0	0	0				

Table C.5
Lot sizes when JIT-philosophy is applied

Part	Selected Route Lot Size Inventory Level in period t = 1	2	3	4	5	6	No of Setups Average Lot Size Average Inventory	No of defective
1	1	-	1	-	1	-	3	
	1057	-	480	-	746	-	761	84
	500	0	80	80	500	0	193	
2	1	1	1	1	1	1	6	
	2404	404	1298	373	1541	404	1071	628
	0	0	0	73	0	0	12	
3	1	-	1	1	1	-	4	
	1399	-	929	423	1176	-	982	429
	0	0	0	0	0	0	0	
4	1	-	1	1	1	-	4	
	4555	-	3166	1677	4144	-	3386	1194
	250	0	0	0	450	0	117	
5	1	-	1	-	-	-	2	
	960	-	1570	-	-	-	1265	31
	860	60	1000	600	600	0	520	
6	2	2	1	2	2	-	5	
	3248	560	5235	604	560	-	2042	749
	0	60	0	0	310	10	63	
7	2	1	2	2	-	-	4	
	2062	560	3839	817	-	-	1820	480
	0	160	0	500	0	0	110	
8	1	1	-	1	1	-	4	
	480	635	-	480	513	-	527	19
	280	600	0	80	90	0	175	
9	1	1	-	1	1	1	5	
	1171	1618	-	1289	1124	426	1126	525
	0	0	0	0	0	7	1	
10	2	2	2	2	2	1	6	
	1054	1618	512	1054	1359	319	987	520
	0	0	0	0	0	0	0	
11	1	1	-	1	1	-	4	
	684	1172	-	1039	1505	-	1101	184
	0	0	0	0	310	0	52	
12	1	1	1	1	1	1	6	
	2928	4373	1484	3506	4217	1306	2970	1398
	0	0	0	0	0	0	0	

Table C.6
Lot sizes when disruptive-philosophy is applied

Part	Selected Route Lot Size Inventory Level in period t = 1	2	3	4	5	6	No of Setups Average Lot Size Average Inventory	No of defective
1	1	-	-	1	-	-	2	
	1612	-	-	999	-	-	1306	412
	900	400	0	800	500	0	433	
2	1	-	-	1	-	-	2	
	3824	-	-	2587	-	-	3206	612
	600	200	0	400	400	0	267	
3	1	-	-	1	-	-	2	
	2195	-	-	1721	-	-	1959	418
	350	350	0	320	0	0	170	
4	1	-	-	1	-	-	2	
	7081	-	-	5826	-	-	6454	558
	650	400	0	650	450	0	358	
5	1	-	-	1	-	-	2	
	1696	-	-	1191	-	-	1444	389
	1400	600	0	600	600	0	533	
6	1	-	-	1	-	-	2	
	5591	-	-	4418	-	-	5005	560
	700	200	0	550	300	0	292	
7	1	-	-	1	-	-	2	
	4295	-	-	3071	-	-	3684	567
	1000	600	0	500	0	0	350	
8	1	-	-	1	-	1	3	
	1306	-	-	1357	-	30	898	594
	900	600	0	750	250	0	417	
9	1	-	-	1	-	-	2	
	2824	-	-	2927	-	-	2876	653
	200	0	0	100	100	0	67	
10	1	-	-	1	-	-	2	
	3175	-	-	2762	-	-	2969	538
	700	500	0	200	0	0	233	
11	3	-	-	3	-	-	2	
	2051	-	-	2683	-	-	2367	515
	500	0	0	750	150	0	233	
12	1	-	-	1	-	-	2	
	8510	-	-	8734	-	-	8622	825
	400	400	0	600	600	0	333	

Bibliography

[1] Adil, G., Rajamani, D., and Strong, D., 1997. Assignment allocation and simulated annealing algorithms for cell formation. *IIE Transactions*, **29**, 53–57.

[2] Akturk, M. and Turkcan, A., 2000. Cellular manufacturing system design using a holonistic approach. *International Journal of Production Research*, **38**, 2327–2347.

[3] Alba, E. and Troya, J. M., 2000. Influence of the migration policy in parallel distributed gas with structured and panmictic populations. *Applied Intelligence*, **12**, 163–181.

[4] Anderberg, M. R., 1973. Cluster Analysis for Applications. Academic Press, New York,

[5] Arvindh, B. and Irani, S., 1994. Cell formation: the need for an integrated solution of the subproblems. *International Journal of Production Research*, **32**, 1197–1218.

[6] Askin, R. G. and Chiu, K. S., 1990. A graph partitioning procedure for machine assignment and cell formation in group technology. *International Journal of Production Economics*, **28**, 1555–1572.

[7] Askin, R. J. and Vakharia, A. J., 1990. Group Technology - Cell Formation and Operation. In Automated Factory Handbook: Technology and Management, D.I. Cleland and B. Bidanda, University of Pittsburgh (eds). TAB Books, Inc., New York, pp. 317–366.

[8] Asokan, P., Prabhakaran, G., and Kumar, G., 2001. Machine-cell grouping in cellular manufacturing systems using non-traditional optimisation techniques a comparative study. *Integrated Manufacturing Systems*, **18**, 140–147.

[9] Awwal, A. A. S. and Karim, M. A., 1989. Machine parts recognition using a trinary associative memory. *Optical Engineering*, **28**, 537–543.

[10] Azencott, R., 1992. Parallel simulated annealing: an overview of basic techniques. In Robert Azencott, editor, Simulated Annealing: Parallelization Techniques. John Wiley and Sons, New York, pp. 25–35.

[11] Azencott, R., 1992. Sequential simulatd annealing: speed of convergence and acceleration techniques. In Robert Azencott, editor, Simulated Annealing: Parallelization Techniques. John Wiley and Sons, New York, pp. 1–9.

[12] Balakrishnan, J. and Jog, P. D., 1995. Manufacturing cell formation using similarity coefficients and a parallel genetic tsp algorithm: Formulation and comparison. *Mathematical and Computer Modelling*, **21 (12)**, 61–73.

[13] Bard, J. and Golany, B., 1991. Determining the number of kanbans in a multiproduct multistage production system. *International Journal of Production Research*, **29**, 881–895.

[14] Baykasoglu, A., Gindy, N., and Cobb, R., 2001. Capability based formulation

and solution of multiple objective cell formation problems using simulated annealing. *Integrated Manufacturing Systems*, **12**, 258–274.

[15] Bitran, G. and Chang, L., 1987. A mathematical programming approach to deterministic kanban system. *Management Science*, **33**, 427–441.

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

[17] Burbridge, J., 1989. Production Flow Analysis for Planning Group Technology. Oxford Science Publication, Oxford,

[18] Burke, L. I. and Kamal, S., 1992. Fuzzy ART and cellular manufacturing. In Artificial Neural Networks in Engeneeing (ANNIE 92). St. Louis, pp. 779–784.

[19] Cabera-Rios, M., Mount-Campbell, C., and Irani, S., 2002. An approach to the desing of a manufacturing cell under economic consideration. *International Journal of Production Economics*, **78**, 223–237.

[20] Cao, D. and Chen, M., 2004. Using penalty function and tabu search to solve cell formation problems with fixed cell cost. *Computers & Operations Research*, **31**, 21–37.

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

[22] Caux, C., Bruniaux, R., and Pierreval, H., 2000. Cell formation with alternative process plans and machine capacity constraints: A new combined approach. *International Journal of Production Economics*, **64**, 179–284.

[23] Caux, C. and Pierreval, H., 1995. Regroupement de machines et doperateurs en cellules par des algorithmes evolutionnistes. In Proceedings of the International Industrial Engineering Conference. Montreal, Canada, pp. 163–172.

[24] Chan, H. and Milner, D., 1982. Direct clustering algorithm for group formation in cellular manufaturing. *Journal of Manufacturing Systems*, **1**, 65–74.

[25] Chandrasekharan, M. P. and Rajagopalan, R., 1986. An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. *International Journal of Production Research*, **24**, 451–464.

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

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

[28] Chen, J. and Heragu, S., 1999. Stepwise decomposition approaches for large scale cell formation problems. *European Journal of Operational Research*, **113**, 64–79.

[29] Chen, M., 1998. A mathematical programming model for system reconfiguration in a dynamic cellular manufacturing environment. *Annals of Operations Research*, **74**, 109–128.

[30] Chen, M., 2001. A model for integrated production planning in cellular manufacturing systems. *Integrated Manufacturing Systems*, **12**, 275–284.

[31] Chen, M. and Cao, D., 2004. Coordinating production planning in cellular manufacturing environment using tabu search. *Computers & Industrial Engineering*, **46**, 571–588.

[32] Chen, S. J. and Cheng, C. S., 1995. A neural network-based cell formation algorithm in cellular manufacturing. *International Journal of Production Research*, **32**, 293–318.

[33] Cheng, C. H., Gupta, Y. P., Lee, W. H., and Wong, K. F., 1998. A tsp-based heuristic for forming machine groups and part families. *International Journal of Production Research*, **36**, 1325–1338.

[34] Chiang, T. S. and Chow, Y., 1988. On the convergence rate of annealing processes. *SIAM journal on control and optimization*, **26**, 1455–1470.

[35] Choobineh, F. A., 1988. A framework for the design of cellular manufacturing systems. *International Journal of Production Research*, **26**, 1161–1172.

[36] Chu, C.-H., 1993. Manufacruing cell formation by comptetive learing. *International Journal of Operations Research*, **31**, 829–843.

[37] Chu, C.-H. and Tsai, M., 1990. A comparison of three array-based clustering techniques for manuacturing cell formation. *International Journal of Operations Research*, **28**, 1417–1433.

[38] Dagli, C. H., 1994. Artificial Neural Network for Intelligent Manufacturing. Chapman & Hall, London,

[39] Dagli, C. H. and Sen, C. F., 1992. Art1 neural network approach to large scale goput technology problem. *ASME Press, New York*, **4**, 787–792.

[40] Deutsch, S. J., Freeman, S. F., and Helander, M., 1998. Manufacturing cell formation using an improved p-median model. *Computers & Industrial Engineering*, **34**, 135–146.

[41] Diaz, a. L. S., B.Z., Racero, J., and Gurrero, F., 2001. Machine cell formation in generalized group technology. *Computers & Industrial Engineering*, **41**, 227–240.

[42] Dutta, S. P., Lashkari, R. S., Nadoli, G., and Ravi, T. A., 1986. A heuristic procedure for determining manufacturing families from design-based grouping for flexible manufacturing systems. *Computers & Industrial Engineering*, **10**, 193–201.

[43] Faber, Z. and Carter, M. W., 1986. A new grph theory approach for forming machine cells in cellular production systems. In Flexible Manufacturing Systems: Methods and Studies, ed. A. Kusiak. North-Holland, New York, pp. pp. 301–318.

[44] Garvin, D. A., 1988. Managing quality: The strategic and competitive edge. Free Press, New York,

[45] Goldberg, D., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York,

[46] Gordon, V. and Whitley, D., 1993. Serial and parallel genetic algorithms as function optimizers. Proceedings of the Fifth International Conference on Genetic Algorithms, edited by S. Forrest. Morgan Kaufmann, San Mateo, CA, pp. 177–183.

[47] Greene, T. and Sadowski, R., 1984. A review of cellular manufacturing assumptions, advantages, and design techniques. *Journal of Operations Management*, **4**, 85–97.

[48] Gunasingh, K. R. and Lashkari, R. S., 1989. The cell formation problem in cellular manufacturing systems–a sequential modelling approach. *Computers & Industrial Engineering*, , 469–471.

[49] Güngör, Z. and Arikan, F., 2000. Application of fuzzy decision making in part-machine grouping. *International Journal of Production Econcomics*, **63**, 181–193.

[50] Gupta, T., 1993. Design of manufacturing cells for flexible environment considering alternative routing. *International Journal of Production Research*, **31**, 1259–1273.

[51] Gupta, T. and Seifoddini, H., 1990. Production data based similarity coefficient for machine componenet grouping in decisions in the desing of a cellular manufacturing systems. *International Journal of Production Research*, **28**, 1247–1269.

[52] Harhallakis, G., Ioannou, G., Minis, I., and Nagi, R., 1994. Manufacturing cell formation under random product demand. *International Journal of Production Research*, **32**, 47–64.

[53] Heragu, S. and Chen, J., 1998. Optimal solution of cellular manufacturing system design: Benders decomposition approach. *European Journal of Operational Research*, **107**, 175–192.

[54] Hertz, A., Jaumard, B., and Ribeiro, C. C., 1994. A graph theory approach to subcontracting, machine duplication and intercell moves in cellular manufacturing. *Discrete Applied Mathematics*, **50**, 255–265.

[55] Holland, J. H., 1975. Adaptation in natural and artificial systems. The University of Press, Michigan,

[56] Hong, Z., Wang, H., and Chen, W., 2000. Simultaneously applying multiple mutation operators in genetic algorithms. *Journal of Heuristics*, **6**, 439–455.

[57] Huge, E. C. and Anderson, A. D., 1988. The Spirit of Manufacturing Excellence: An Executives Guide to the New Mind Set. Dow Jones-Irwin, Homewood, ILL,

[58] Hyer, N. and Wemmerlöv, U., 2002. REORGANIZING THE FACTORY Computing through cellular manufacturing. Productivity Press, Portland, Oregon, USA, pp. 311–368.

[59] Hyer, N. L. and Wemmerlöv, U., 1989. Groupt technology in the us manufacturing industry: A survey of current practice. *International Journal of Production Research*, **27**, 1287–1304.

[60] ILOG Inc., 2003. CPLEX 9.0 Users Manual. 1080 Linda Vista Ave. Mountain View, CA 94043, (http://www.ilog.com).

[61] Islam, K. M. S. and Sarker, B. R., 2000. A similarity coefficient measure and machine-part grouping in cellular manufacturing systems. *International Journal of Production Research*, **38**, 699–720.

[62] Jaber, M. and Bonney, M., 2003. Lot sizing with learning and forgetting in set-ups and in product quality. *International Journal of Production Economics*, **83**, 95–111.

[63] Jacobs, F. and Bragg, D. J., 1996. Repetitive lots: flow time reduction through sequencing and dynamic batch sizing. *Decision Science*, **19**, 281–294.

[64] Jamal, A., 1993. Neural network and cellular manufacturing. *Industrial Management & Data Systmes*, **39**, 21–25.

[65] Jeon, G., Leep, H. R., Hamid, R., and Parsaei, H. R., 1998. A cellular manufacturing system based on new similarity coefficient which considers alternative routes during machine failure. *Computers & Industrial Engineering*, **34**, 21–36.

[66] Joines, J., Culbreth, and C., R., King, 1996. A comprehensive review of production oriented cell formation techniques. *International Journal of Factory Automation and Information Management*, **3**, 225–265.

[67] Josien, K. and Liao, T. W. R., 2002. Simultaneous grouping of parts and machines with an integrated fuzzy clustering method. *Fuzzy Sets and Systems*, **126**, 1–21.

[68] Kamrani, A. and Parsaie, H., 1993. A group technology based methodology for machine cell formation in a compu- ter integrated manufacturing environment. *Computers & Industrial Engineering*, **24**, 431–447.

[69] Kamrani, A., Parsaie, H., and Chaudhry, M., 1993. A survey of design methods for manu- facturing cells. *Computers & Industrial Engineering*, **25**, 487–490.

[70] Kaparthi, S. and Suresh, N., 1991. A new network system for shape-based classification and coding of rotational parts. *Interantional Journal of Production Research*, **29**, 1771–1784.

[71] Kaparthi, S. and Suresh, N., 1992. Machine-component cell formation in goup technology: a neural network approach. *Interantional Journal of Production Research*, **30**, 1353–1367.

[72] Kaparthi, S. and Suresh, N., 1994. Performance of selected part-machine grouping techniques for data sets of wise ranging sizes and imperfection. *Decision Sciences*, **25**, 515–539.

[73] Kazerooni, L. M., Luong, H. S., and Kazem, A., 1997. A genetic algorithm based cell design considering alternative routing. *Computer Integrated Manufacturing Systems*, **10**, 93–107.

[74] Khator, S. and Irani, S., 1987. Cell formation in group technology: A new approach. *Computers & Industrial Engineering*, **12**, 131–142.

[75] Kim, Y.-D., 1993. A study on surrogate objectives for loading a certain type of flexible manufacturing system. *International Journal of Production Research*, **31**, 381–392.

[76] King, J., 1980. Machine-component grouping in pfa: an approach using a rank-order clustering algorithm. *International Journal of Production Research*, **18**, 213–232.

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

[78] King, R. E., Joines, J. A., and Culbreth, C. T., 1995. An intelligent, genetic search for the cell formation problem. In the Proceedings of the International Industrial Engineering Conference. Montreal, Canada, pp. 245–254.

[79] Kirkpatrick, S., Gelatt, C., and Vecchi, M., 1983. Optimization by simulated annealing. *Science*, **220**, 671–680.

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

[81] Kusiak, A., 1988. Exgt-s: A knowledge based system for group technology. *International Journal of Production Research*, **26**, 887–904.

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

[83] Kusiak, A. and Chow, W., 1987. Group technology. *Computers & Industrial Engineering*, **9**, 83–91.

[84] Kusiak, A. and Chung, Y., 1991. Using neural networks to form machine cells. *Manufacturing Review*, **4**, 293–301.

[85] Lee, H. and Garcia-Diaz, A., 1996. Network flow procedures for the analysis of cellular manufacturing systems. *IIE Transactions*, **28**, 333–345.

[86] Lee, S.-Y. and Lee, K. G., 1996. Synchronous and asynchronous parallel simulated annealing with multiple markov chains. *IEEE Transactions on Parallel and Distributed Systems*, **7**, 903–1007.

[87] Lee-Post, A., 2000. Part family identification using a simple genetic algorithm. *International Journal of Production Research*, **38**, 793–810.

[88] Lemoine, Y. and Mutel, B., 1983. Automatic recognition of production cells and part families. In Advances in CAD/ CAM, Ellis, T. M. R., and Semenkov, O. I.,(eds). North-Holland, Amsterdam, pp. 239–248.

[89] Li, C. and Cheng, T., 1994. An economic production quantity model with learning and forgetting considerations. *Production and Operations Management*, **3**, 118–132.

[90] Liao, T. W. and Lee, K. S., 1994. Integration of a feature-based cad system and an art1 neural model for gt coding and part family forming. *Computers & Industrial Engineering*, **26**, 93–104.

[91] Lockwood, W., Mahmoodi, F., and Ruben, R., 2000. Scheduling unbalanced cellular manufacturing systems with lot splitting. *International Journal of Production Research*, **38**, 951–965.

[92] Logendran, R. and Ko, C. S., 1997. Manufacturing cell formation in the presence of flexible cell locations and material transporters. *Computers & Industrial Engineering*, **33**, 445–448.

[93] Logendran, R. and Puvanunt, V., 1997. Duplication of machines and subcontracting of parts in the presence of alternative cell locations. *Computers & Industrial Engineering*, **33**, 235–238.

[94] Logendran, R. and Ramakrishna, P., 1997. A methodology for simultaneously dealing with machine duplication and part subcontracting in cellular manufacturing systems. *Computers & Operations Research*, **24**, 97–116.

[95] Lozano, S., Dobado, D., Larrañeta, J., and Onieva, L., 2002. Modified fuzzy

c-means algorithm for cellular manufacturing. *Fuzzy Sets and Systems*, **126**, 23–32.

[96] Luong, H. J. A. K., L., 2002. A decision support system for cellular manufacturing system desing. *Computers & Industrial Engineering*, **42**, 457–470.

[97] Malave, C. O., 1991. Aneural network-based desing of cellular manufacturing systems. *Jornal of Intelligient Manufacturing*, **2**, 305–314.

[98] McAuley, J., 1972. Machine grouping for efficient production. *Production Engineering*, **51**, 53–57.

[99] McCromic, W., Schweitzer, P., and White, T., 1972. Problem decomposition and data reorganization by a cluster technique. *Operations Research*, **20**, 993–1009.

[100] Meise, C., 1998. On the convergence of parallel simulated annealing. *Stochastic Processes and their Applications*, **76**, 99–115.

[101] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., 1953. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, **21**, 1087–1092.

[102] Michalewicz, Z., 1994. Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York,

[103] Monden, Y., 1983. Toyota Production Systems. Industrial Engineering and Management Press, Norcross, GA,

[104] Moon, Y. B., 1990. An interactive and comtetion model for machine-part family formation in group technology. an international joint conference on neural

networks. Proceedings of the International Industrial Engineering Conference. Washington, D.C., pp. 667–670.

[105] Mosier, C. T., 1989. An experiment investigating the application of clusterin procedures and similarity coeffeicients to the gt malchine cell formation problem. *International Journal of Production Research*, **27**, 1811–1835.

[106] Mukhopadhyay, S. K., Babu, K. R., and Vijai Sai, K., 2000. Modified hamiltonian chain: a graph theoretic approach to group technology. *International Journal of Production Research*, **38**, 2459–2470.

[107] Mungwattana, A., 2000. Design of cellular manufacturing systems for dynamic and uncertain production requirements with presence of routing flexibility. Ph.D. thesis, Virginia Polytechnic Institute and State University, Blackburg, VA.

[108] Nair, G. J. and Narendran, T. T., 1998. Case: A clusterin algorithm for cell formation with sequence data. *International Journal of Production Research*, **36**, 157–179.

[109] Narayanaswamy, P., Bector, C. R., and Rajamani, D., 1996. Fuzzy logic concepts applied to machine-component matrix formation in cellular manufacturing. *European Joumal of Operational Research*, **98**, 88–97.

[110] Nowostawski, M. and Poli, R., 1999. Parallel genetic algorithm taxonomy. Proceedings of the Third International Conference on Knowledge-Based Intelligent Information Engineering System. Adelaide, Australia, pp. 88–92.

[111] Ohta, H. and Nakamura, M., 2002. Cell formation with redution in setup times. *Computers & Industrial Engineering*, **42**, 317–327.

[112] Olorunniwo, F., 1997. A framework for measuring success of cellular manufacturing implementation. *International Journal of Production Research*, **35**, 3043–3061.

[113] Onwubolu, G. and Mutingi, M., 2001. A genetic algorithm approach to cellula manufacturing systems. *Computers & Industrial Engineering*, **39**, 125–144.

[114] Pierreval, H., Caux, C., Paris, J. L., and Viguier, F., 2003. Evolutionary approaches to the design and organization of manufacturing systems. *Computers & Industrial Engineering*, **44**, 339–364.

[115] Pierreval, H. and Plaquin, M. F., 1994. An evolutionary approach to multicriteria cell formation problems. *International Transactions in Operational Research*, **5**, 13–25.

[116] Pierreval, H. and Plaquin, M. F., 1994. A genetic algorithm approach to group machines into manufacturing cells. Fourth International Conference on Computer Integrated Manufacturing and Automation Technology. Troy, New York, pp. 267–271.

[117] Plaquin, M.-F. and Pierreval, H., 2000. Cell formation using evolutionary algorithms with certain constraints. *International Journal of Production Economics*, **64**, 267–278.

[118] Porteus, E. L., 1986. Optimal lot sizing, process quality improvement and setup cost reduction. *Operations Research*, **34**, 137–144.

[119] Price, W., Gravel, M., and Nsakanda, A., 1994. A review of optimization models of kanaban-based production systems. *Journal of Operational Research*, **75**, 1–12.

[120] Price, W., Gravel, M., Nsakanda, A., and Cantin, F., 1995. Modeling th eperformance of a kanban assembly shop. *International Journal of Production Research*, **33**, 1171–1177.

[121] Purcheck, G., 1974. A mathematical classification as a basis for the desing of group technology production cells. *Production Engineer*, **54**, 35–48.

[122] Purcheck, G., 1975. A linear programming method for the combinatorial grouping of an incomplete power set. *Journal of Cybernetics*, **5**, 51–76.

[123] Rajagopalan, R. and Batra, J., 1975. Design of cellular production systems–a graph theoretic approach. *International Journal of Production Research*, **13**, 56–68.

[124] Rajamani, D., Singh, N., and Aneja, Y., 1990. Integrated desing of cellular manufacturin systems in the presence of alternative prcess plans. *International Journal of Production Research*, **30**, 1541–1554.

[125] Rao, H. A., Pham, S. N., and Gu, P., 1999. A genetic algorithms-based approach for design of manufacturing systems: An industrial application. *International Journal of Production Research*, **37**, 557–580.

[126] Ribeiro, J. F. F. and Pradin, B., 1993. A methodology for cellular manufacturing design. *International Journal of Production Research*, **31**, 235–250.

[127] Riggs, J., 1981. Production Systems: Planning, analysis and control. Wiley, New York,

[128] Schonberger, R. J. and Knod, E. M., 1994. Operations Management: Continuous Improvement, 5th Edition. Irwin, Burr Ridge, IL,

[129] Seifoddini, H., 1989. Single linkage versus average linkage clustering in machine cell formation applications. *Computers & Industrial Engineering*, **16**, 419–426.

[130] Seifoddini, H., 1990. A probabilistic model for machine cell formation. *International Journal of Production Research*, **9**, 69–75.

[131] Selim, H., Askin, R., and Vakharia, A., 1998. Cell formation in groupt technology: Review, evaluation and direction for future research. *Computers & Industrial Engineering*, **34**, 2–30.

[132] Selvam, R. P. and Balasubramanian, K. N., 1985. Algorithmic grouping of operation sequences. *Engineering Costs and Production Economics*, **9**, 125–134.

[133] Shtub, A., 1989. Modeling group technology cell formation as a generalized assignment problem. *International Journal of Production Research*, **27**, 775–782.

[134] Singh, N., 1996. Systems Approach to Computer-Integrated Design and Manufacturing. Wiley, New York,

[135] Sofianopoulou, S., 1997. Application of simulated annealing to a linear model for the formulation of machine cells in group technology. *International Journal of Production Research*, **32**, 501–511.

[136] Sofianopoulou, S., 1999. Manufacturing cells design with alternative process plans and/or replicate machines. *International Journal of Production Research*, **37**, 707–720.

[137] Solimanpur, M., Vrat, P., and Shankar, R., 2004. A multi-objective genetic

algorithm approach to the design of cellular manufacturing systems. *International Journal of Production Research*, **42**, 1419–1441.

[138] Spearman, M. and Zaranis, M., 1992. Push and pull prouduction systems: issues and comparisons. *Operations Research*, **40**, 521–532.

[139] Srinivasan, G., 1994. A clusterin algorithm for cell formation in group technology using minimum spnning trees. *International Journal of Production Research*, **32**, 2140–2158.

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

[141] Srinivasan, G., Narendran, T., and Mahadevan, B., 1990. An assignment model for the part families problem in group technology. *International Journal of Production Research*, **28**, 145–152.

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

[143] Su, C.-T. and Hsu, C.-M., 1998. Multi-objective machine-part cell formation through parallel simulated annealing. *International Journal of Production Research*, **36**, 2185–2207.

[144] Süer, G., Saiz, M., and González, W., 1999. Evaluation of manufacturing cell loading rules for independent cells. *International Journal of Production Research*, **36**, 2185–2207.

[145] Suresh, N. C. and Kaparthi, S., 1994. Performacne of fuzzy art neurla network for group technology cell formation. *International Journal of Production Research*, **32**, 1693–1713.

[146] Tam, K. Y., 1990. An operation sequence based similarity coefficient for part families formations. *Journal of Manufacturing Systems*, **9**, 55–68.

[147] Urban, T. L., 1998. Analysis of production systems when run length influences product quality. *International Journal of Production Research*, **36**, 3085–3094.

[148] Vakharia, A. J. and Slim, H., 1994. Group Technology. In R.C. Dorf and A. Kusiak (eds), The Handbook of Design, Manufacturing, and Automation. Wiley, New York, pp. 435–460.

[149] Vakharia, A. J. and Wemmerlöve, U., 1994. Designing a cellular manufacturing system: a materials flow approach based on operation sequences. *IIE Transactions*, **22**, 84–97.

[150] Venugopal, A. and Narendran, T. T., 1992. A neural network approach for desinging cellular manufactuing systems. *Advances in Modeling and Analysis*, **32**, 13–26.

[151] Viguier, F. and Pierreval, H., 2001. Structuring hybrid manufacturing systems. Proceedings of the Industrial Engineering and Production Management IEPM Conference. Quebec, Canada, pp. 931–940.

[152] Viswanathan, S., 1996. A new approach for solving the p-median problem in group technology. *International Journal of Production Research*, **34**, 2691–2700.

[153] Vohra, T., Chen, D., Chang, J., and Chen, H., 1990. A network approach to cell formation in celluar manufacturing. *International Journal of Production Research*, **28**, 2075–2084.

[154] Wagner, B. and Ragatz, G. L., 1994. The impact of lot splitting on due date performance. *Journal of Operations Management*, **12**, 13–25.

[155] Wang, J., 2003. Formation of machine cells and part families in cellular manufacturing systems using a linear assignment algorithm. *Automatica*, **26**, 1607–1615.

[156] Wang, J. and Roze, C., 1997. Formation of machine cells and part families in cellular manufacturing: an experimental study. *Journal of Operations Management*, **35**, 1259–1286.

[157] Wang, T., Wu, K. B., and Liu, Y. W., 2001. A simulated annealing algorithm for facility layout problems under variable demand in cellular manufacturing. *Computers in Industry*, **46**, 181–188.

[158] Wemmerlöv, U. and Hyer, N. L., 1986. Procedures for part-family/machine group identification problem in cellular manufacturing. *Journal of Operations Management*, **6**, 125–145.

[159] Wemmerlöv, U. and Johnson, D., 1997. Cellular manufacturing at 46 user plants: implementation experiences and performance improvements. *International Journal of Production Research*, **35**, 29–49.

[160] Wicks, E. M. and Reasor, R. J., 1999. Designing cellular manufacturing systems with dynamic part populations. *IIE Transactions*, **31**, 11–20.

[161] Witte, J. D., 1980. The use of similartiy coefficients inproduction flow analysis. *International Journal of Production Research*, **18**, 502–514.

[162] Won, Y., 2000. Two-phase approach to gt cell formation using efficient p-median formulations. *International Journal of Production Research*, **38**, 1601–1613.

[163] Won, Y. and Lee, K. C., 2004. Modified p-median approach for efficient gt cell formation. *Computers & Industrian Engineering*, **46**, 495–510.

[164] Wu, H., Venugopal, R., and Barash, M., 1986. Design of a cellular manufacturing system: A syntactic pattern recognition approach. *Journal of Manufacturing Systems*, **5**, 81–88.

[165] Xambre, A. R. and Vilarinho, P. M., 2003. A simulated annealing approach for manufacturing cell formation with multiple identical machines. *European Journal of Operational Research*, **151**, 434–446.

[166] Xu, H. and Wang, H. P., 1989. Part family formation for gt applications based on fuzzy mathematics. *International Journal of Production Research*, **27**, 1637–1651.

[167] Yasuda, K. and Yin, Y., 2003. A dissimilarity measure for solving the cell formation problem in cellular manufacturing. *Computers & Industrial Engineeing*, **39**, 1–7.

[168] Zhang, C. and Wang, H., 1992. Concurrent formation of part families and machine cells based on the fuzzy set theory. *Journal of Manufacturing Systems*, **11**, 61–67.

[169] Zhang, H.-C. and Huang, S. H., 1995. Applications of neural networks in manufacturing: A state-of-the-art survey. *International Journal of Production Research*, **33**, 705–728.

[170] Zhao, C. and Zhiming, W., 2000. A genetic algorithm for manufacturing cell formation with multiple routes and multiple objectives. *International Journal of Production Research*, **38**, 385–395.

[171] Zolfaghari, S. and Liang, M., 2003. A new genetic algorithm for the machine/part grouping problem involving processing times and lot sizes. *Computers & Industrial Engineering*, **45**, 713–731.