INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Université d'Ottawa · University of Ottawa

Design and Planning for Cellular Manufacturing: Application of Neural Networks and Advanced Search Techniques

by

Saeed Zolfaghari

A Thesis Submitted to the School of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of

Doctor of Philosophy
in
Mechanical Engineering

Ottawa-Carleton Institute for
Mechanical and Aerospace Engineering

University of Ottawa
Ottawa, Ontario, Canada

Dedicated to

my wife, Shila, for her patience

and

my lovely daughter, Nika, for her understanding

ABSTRACT

A cellular manufacturing system (CMS) is a manufacturing structure organized based on the group technology (GT) concept. The main advantages of the CMSs include the low material handling costs, short setup times and reduced work in process. This study addresses the machine/part grouping and group scheduling (i.e., part/part family scheduling) problems, the two key issues in the CMS design and planning.

The machine/part grouping problems can be classified into binary and comprehensive grouping problems depending on whether or not the processing times and the machine capacities are considered. The binary grouping problem arises if the part demands are unknown when the CMS is being developed. If the part demand can be forecast accurately, both the processing times and machine capacities have to be included in the analysis. This gives rise to comprehensive grouping. Both the binary and comprehensive grouping have been proved to be NP-complete problems which cannot be solved in polynomial time. Considering the large number of parts and machines involved in the industrial design problem, efficient solution methods are highly desirable.

In this study, a novel neural network structure, Ortho-Synapse Hopfield Network (OSHN), has been designed to solve the binary grouping problem. Due to its significantly reduced number of synapses and unique structure, the OSHN is very computationally efficient and training-free. An objective-guided search approach has been developed to lead the OSHN search process to tune the network parameters and escape the local optima. To solve the comprehensive grouping problem, two approaches are proposed. The first one is a simulated annealing (SA) method based on a generalized grouping efficiency index. The SA method is used jointly with the OSHN algorithm to improve

the computational efficiency. The second method is a modified OSHN algorithm. The objective of the modified OSHN is to maximize the generalized grouping efficiency subject to machine capacities. Our computational results compare favorably with solutions obtained in the literature.

The group scheduling problem has also been proven to be NP-hard. Furthermore, due to the limited time available for scheduling decisions, computational efficiency is more critical. To this end, a combined tabu search/simulated annealing (tabu-SA) approach is developed to solve the group scheduling problem. The main advantage of this approach is that the simulated annealing search can be accompanied by a short term memory to avoid cycling and thus improve solution quality and computational efficiency. This has been tested and demonstrated in our computational experience.

ACKNOWLEDGEMENTS

I wish to extend my deepest appreciation to my supervisor, Dr. M. Liang, for his innumerable suggestions, time, insight, support and continuous guidance throughout this study. I am also grateful to all members of my defense committee for their effort and time spent on this thesis.

The financial support provided by the Ministry of Culture and Higher Education of Iran is greatly appreciated. My special appreciation goes out to my parents for their encouragement and moral support.

Last but not least, I am deeply indebted to a wonderful, supportive and considerate family, particularly my wife, Shila, who sacrificed her own comfort over the many years so that I could pursue what she felt was important to me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

x,y,i	index of machines (or cities or tools), $x,y,i=1,\ldots,M$
$[i]$	index of order of operations, $[i]=1,\ldots,M$
j,g	index of parts, $j,g=1,\ldots,N$
f	index of positions of families in the family-sequence, $f=1,\ldots,R$
k,l	index of positions of parts in the part-sequence, $k,l=1,\ldots,N$
r,q	index of cells and part families (or positions in a tour in TSP), $r,q=1,\ldots,R$
s,v	index of sites, $s,v=1,\ldots,S$
t	index of iterations
a_{xy}	number of parts that require operations on both machines x and y
b_{xy}	number of parts that require machine x, but not machine y
c_{xy}	number of parts that require machine y, but not machine x
C_{rj}	completion time for part j of family r
C_{rij}	completion time for part j of part family r on machine i
C^T	one-minute tool life machining speed
$C(X)$	cost of state X
CTR	a counter used in Algorithm 3.4 to record number of additional search steps in the current direction after a worse solution has been detected
d_{rj}	due date of part j of family r
$\bar{d}_r$	average due date of parts in family r
d_r^*	modified due date for family r
D_{rj}	degree of belongingness of part type j to cell r

A,B,C,D	tuneable parameters of the network
$\Delta A,\Delta C,\Delta D$	increments of parameters A, C, and D respectively
e_d	number of elements in all diagonal blocks
e_o	number of elements outside diagonal blocks (exceptional elements)
E	energy level for Hopfield neural network
F_r	set of machines in cell r
F_{ri}	completion time for family r on machine i
G'	number of machine cells generated in an iteration
G	desired number of machine cells
h_{rj}	processing time of part type j in cell r
H_j	total processing time required by part type j
K	maximum number of allowed machine cells
k^l	labour cost ($/min)
k^m	machining overhead cost ($/min)
k^t	cost of a cutting edge ($/edge)
l_{xy}	distance from city (site) x to city (site) y
L_j	lot size of part type j (or expected demand for part type j)
m	total number of machines (including duplicated machines)
m_j	total number of machine types required by part type j
m_{rj}	number of machines in cell r required by part j
M_r	number of machines in cell r
MA,MC,MD	upper limits of parameters A, C, and D
n	total number of parts, $n = \sum_{j=1}^{N} L_j$

n_x	number of available duplicates of tool x
n^T	slope of tool life curve
N_r	number of parts in family r
NS	number of allowed additional steps in the current direction after a worse solution has been detected
p_r	setup time for family r
P	available machining time
q, q_g	weighting factor, $0 \leq q, q_g \leq 1$
s_{rij}	setup time for part j of part family r on machine i
S_r	set of sites in cell r
S_{xy}	similarity coefficient between machines (tools) x and y
$\hat{S}_{yq}$	average similarity coefficient between machine y and cell q
t_{ij}	processing time of part j on machine type i
t_j^{max}	maximum processing time of part type j, $t_j^{max} = \max_i \{t_{ij}\}$
t_{rij}^{min}	minimum processing time for operation i of part j of part family r
t_{rij}^{max}	maximum processing time for operation i of part j of part family r
t_{max}	maximum allowed number of iterations
t^c	time required to replace a worn cutting tool
T	tool magazine capacity
T_{rj}	tardiness of part j of family r
TT	total weighted tardiness (weighted sum of tardiness of all parts)
u	number of allowed small cells
U_{xr}	state of neuron (x,r), $U_{xr} = 1$, if neuron (x,r) is active; 0, otherwise
v_{rij}	machining speed for part j of part family r on machine i

v_{rij}^{min}	minimum allowed speed for operation i of part j of part family r
v_{rij}^{max}	maximum allowed speed for operation i of part j of part family r
v_{rij}^{c}	minimum-cost speed for operation i of part j of part family r
v_{rij}^{slack}	the lowest possible speed subject to the available slack time for operation i of part j of part family r
V	a parameter used in Algorithm 3.4, $V = M$, if the number of machine cells is not restricted; $V = K$, otherwise.
$w_{xr,yq}$	connection weight from neuron (x,r) to neuron (y,q) in the application of the original Hopfield network to GT problem
$W_{xr,yq}$	connection weights from neuron (x,r) to neuron (y,q) in the travelling salesman problem
x_{rijk}	1, if the operation of part j of family r is in position k in the sequence of machine i; 0, otherwise
y_{rf}	1, if part family r is in position f in the family-sequence; 0, otherwise
z_{jxy}	1, if part type j is to be processed by machine x and followed by machine y; 0, otherwise
α	a uniform random number from the interval [0,1]
β_{rj}	penalty cost for each unit of time that part j of family r is tardy
Λ	a machine set in which each member has been assigned to more than one cell
δ_{xy}	Kronecker delta function, $\delta_{xy} = 1$, if $x=y$; 0, otherwise
ϵ	a threshold number of unassigned machines
η	grouping efficiency
η_1	in-cell density of 1's

η_2	out-cell density of 0's
η_g	generalized grouping efficiency
η_d	density factor of processing times in diagonal blocks
η_o	sparsity factor of processing times outside diagonal blocks
φ	grouping efficiency obtained in an iteration
φ_g	generalized grouping efficiency obtained in an iteration
Γ	a machine set which consists of all machines
ρ	set of machines that have been assigned to machine cells and do not belong to Λ
Φ	set of machines to be grouped in the current trial
Ψ	set of unassigned machines
$\omega_{xr,yq}$	connection weight from neuron (x,r) to neuron (y,q) in the application of ortho-synapse Hopfield network to GT problem
Ω_r	set of parts assigned to cell r
τ_{yq}	threshold value for Hopfield neural network
λ	workload ratio, the ratio of in-cell workload to out-cell workload
λ_{rij}	machining constant for operation i of part j of part family r
θ_t	temperature at iteration t
β_t	cost of current state at iteration t
γ_t	cost of neighbouring state at iteration t
π_t	cost of optimal state at iteration t

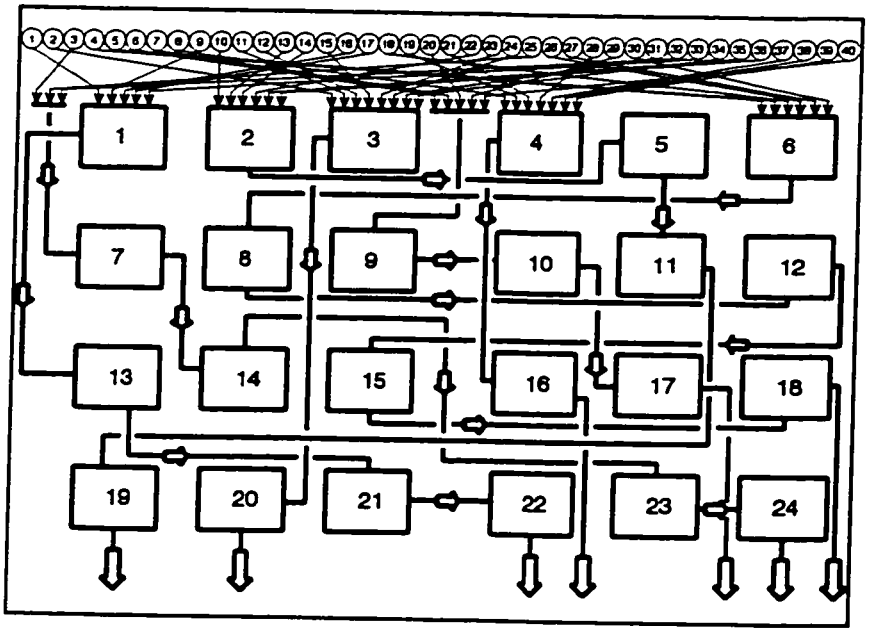
CHAPTER 1

INTRODUCTION

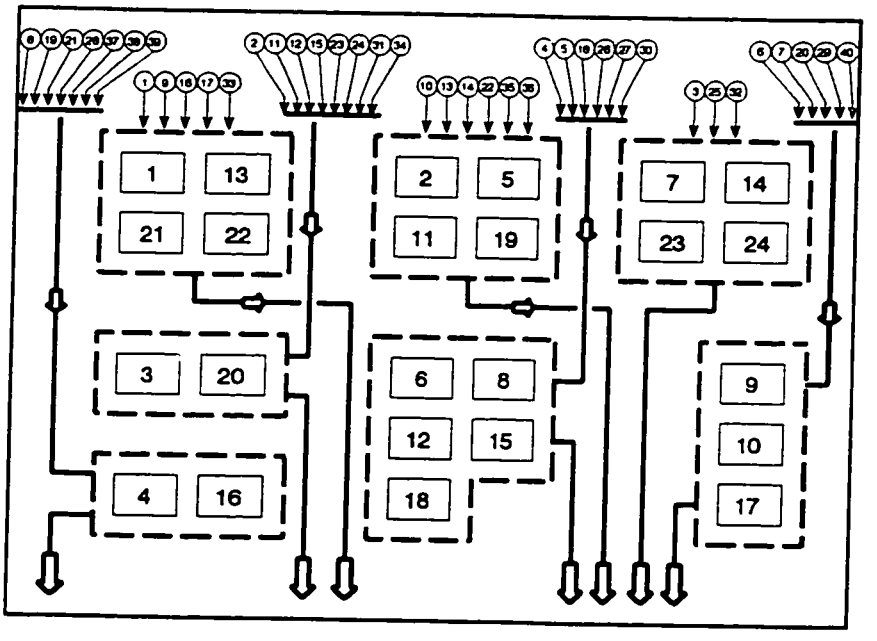
1.1 Overview

Group technology (GT) is a manufacturing philosophy that takes advantage of similarities among the system components to improve the productivity of batch-type manufacturing systems. A manufacturing system based on GT concept is known as the *cellular manufacturing system* (CMS). Though flow shop layout has high efficiency in mass production, and job shop layout has the advantage of high flexibility when a variety of parts are involved, each of them lacks the advantages of the other for small to medium batch production. Cellular manufacturing is a compromise between flow line and job shop production aiming at higher flexibility and efficiency for small to medium batch production. In a cellular manufacturing system, parts are classified into *part families* based on the similarities of their process requirements, and machines are grouped into *machine cells* based on the operation requirements of the part families. In practice, each machine cell is dedicated to one or more part families. A comparison between a job shop system and a cellular manufacturing system is shown in Figure 1.1.

The most appealing benefits of cellular manufacturing are the lower material handling costs, reduced work-in-process and shorter setup times. Achieving these benefits depends on how the machines and parts are grouped. Ideally, fully utilized independent cells are sought in a cellular manufacturing. Every such cell has two characteristics:



(a) Material flow in a job shop



(b) Material flow in a cellular manufacturing system

Figure 1.1. Comparison between a job shop system and a cellular manufacturing system

a) it is fully utilized, i.e., all machines in the cell are required by every part in the family; b) the cell is independent, i.e., each part in the family is completely processed in the cell without requiring any process from another cell. In reality, however, such a grouping result may never be achieved because different parts usually have different process requirements. In other words, intercell movement and under-utilized cells are often inevitable but could be reduced or minimized.

The *binary* grouping problem, in which all operations are treated equally, is a hard optimization problem and has been shown to be NP-complete. That means, the computational times increase at a non-polynomial rate with problem sizes (King 1980). Complete enumeration of all feasible clustering alternatives is impossible for real problems. The number of alternatives in which M elements can be grouped into R non-empty clusters is given by Stirling's number, $S(M,R) \approx R^M/R!$ (Duda and Hart 1973). As Lee and Garcia-Diaz (1993) pointed out, this number could be much greater when the number of clusters is unknown. In that case, the number of clusters could be any number between 1 and M. As a result, the total number of alternatives for a complete enumeration will be $\sum_{r=1}^{M} r^M/r!$ that is an astronomical number even for a relatively small problem (e.g., more than 4×10^{18} for a problem of 25 elements). For this reason, heuristics have been widely used to solve this problem.

The binary based approaches aim at maximizing the machine similarities in each cell. This yields reduced setup times and material handling costs. Nevertheless, these approaches are based on the following assumptions (Venugopal and Narendran 1992):

a) The processing times of different part types are identical for all machines.

b) The lot sizes of all part types are equal.

These assumptions could be valid at early design stage when the demands for different

products are usually unknown. However, if the product demands can be accurately forecast, then more practical parameters such as processing times and machine capacities should be considered. To deal with machine capacity limitations, different policies have been proposed in the literature. These policies are: 1) bottleneck machine duplication, 2) exceptional part subcontracting, 3) multiple process plans, and 4) machine duplication coupled with lot splitting. Among these policies, the machine duplication is most widely used in the literature.

A *comprehensive* grouping problem that incorporates these parameters is far more complex. In order for a binary-based solution to accommodate the processing times, part demands and machine capacities, further manipulations of the final results are usually required. These manipulations seek improvements in within-cell utilization and work load balance.

Once the machine/part grouping problem has been solved, the planning issues such as scheduling of part families and parts within families become crucial. The traditional approaches in flow shop or job shop scheduling may not be applicable to a cellular manufacturing system. In fact, in the transition from a traditional flow or job shop to a cellular manufacturing system, the total setup time of all parts that form a part family is reduced to a major *family setup time* and some minor *part setup times*. The family setup time consists of all preparation activities that are common for all parts in the family, and the rest of the preparation time that is exclusively for each individual part is called part setup time. Part setup times are small compared to family setup times, and may be included in part processing times. This new setup time structure makes the scheduling of cellular manufacturing systems different from flow or job shop problems. In such a scheduling problem, decisions are often made in two stages, i.e., scheduling part families and scheduling parts within each family. For this reason, some researchers

refer to this type of scheduling as *two-stage scheduling*, while others call it *group scheduling* (Hitomi and Ham 1977), which is adopted in this thesis.

The layout pattern of the machines in a machine cell could be either flow shop or job shop, depending on the nature of the process. The parts within each family could be sequenced according to one of the conventional dispatching rules such as SPT and EDD, but part families may be sequenced using a quite different dispatching rule. The lack of compatibility of conventional rules in group scheduling has encouraged researchers to develop new dispatching rules for group scheduling. DD/SPT (Mahmoodi and Dooley 1991), SPT/SPT (Wemmerlöv 1992) and MJ/SI^x (Frazier 1996) are some examples of these new rules. These rules are generally classified into two groups: *exhaustive* and *non-exhaustive* rules. In an exhaustive rule, once the processing of a part family starts, it cannot be interrupted till all parts of that family are processed. However, in a non-exhaustive rule, the processing of a family may be interrupted by jobs with higher priority from another family. The exhaustive rules seem to be more compatible with the principles of cellular manufacturing systems since they tend to avoid family setups. Furthermore, in a simulation study carried out by Mahmoodi and Dooley (1991), it is shown that exhaustive rules outperform the non-exhaustive rules in terms of both mean flow time and mean tardiness.

In this thesis, both the CM design issues (binary grouping and comprehensive grouping) and CM planning issues, in particular, an exhaustive group scheduling problem with the objective of minimizing the total weighted tardiness, will be investigated.

1.2 Objectives

The objectives of this thesis are to: a) design an efficient neural network structure for solving the binary machine/part grouping problem; b) develop an efficient approach

to solving the comprehensive grouping problem incorporating part demands, machine capacities and processing times; and c) model and efficiently solve the group scheduling problem with controllable machining speeds. The first two objectives are related to CMS design, and the last one is related to CMS planning.

1.3 Organization

The organization of this thesis is as follows. Chapter 2 presents the literature review on the design and planning issues in cellular manufacturing systems. The binary grouping problem will be discussed in Chapter 3, and an objective-guided search approach based on neural networks will be developed for solving this problem. In Chapter 4, the comprehensive grouping problem will be discussed and two solution procedures based on simulated annealing and neural networks will be proposed. The group scheduling problem will be modelled and solved using a tabu search method, a simulated annealing approach, and a tabu-simulated annealing method in Chapter 5. The performance of the three different approaches will be compared. The concluding remarks will be outlined in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

In this chapter, the most prominent studies on cellular manufacturing will be reviewed. The literature is classified in accordance with the three topics addressed in this thesis, i.e., binary machine cell/part family formation, comprehensive machine cell/part family formation, and group scheduling.

2.1 Binary Machine Cell/Part Family Formation

Binary grouping has been extensively studied in the literature over the years. The large number of publications on this topic preclude a comprehensive review. Only the influential studies are reviewed in this thesis.

2.1.1 Heuristics

Similarity coefficient based methods

Burbidge (1963) was among the first who addressed group technology and proposed to use production flow analysis for machine/part grouping. Later, McAuley (1972) introduced a more systematic approach called Single Linkage Cluster Analysis (SLCA) which was originally suggested by Sokal and Sneath (1968). McAuley (1972) applied this approach to the group technology problem using a similarity coefficient. Since then, similarity coefficient methods have attracted much attention. One reason for this as stated by Seifoddini (1989a) is that similarity coefficient methods are more flexible in accommodating the manufacturing data in cell formation process. DeWitte

(1980) advanced this approach by introducing three new similarity coefficients based on interdependencies among machines.

The chaining problem and bottleneck machine duplication issue associated with SLCA was addressed by Seifoddini and Wolfe (1986). To deal with this issue they proposed an Average Linkage Clustering (ALC) algorithm. Seifoddini (1989b) compared the SLCA and ALC and concluded that the application of the former approach is easier and faster, but the solution obtained by the latter is preferable due to the reduced inter-cellular moves. Seifoddini (1989a) also proposed a cost-based duplication procedure to reduce the inter-cellular material handling cost. The chaining problem has also been investigated by Chow and Hawaleshka (1992).

The selection of the threshold value is an important decision in similarity coefficient methods. This threshold value is usually selected arbitrarily. Waghodekar and Sahu (1984) developed a machine-component cell formation algorithm (MACE) based on a similarity coefficient of the product type without arbitrary selection of the threshold value. Seifoddini and Wolfe (1987) dealt with this issue in a different way. They proposed to determine the threshold value based on a cost function for inter-cellular material handling.

Rearranging the machine-part matrix

Another approach to machine/part grouping is rearranging rows and columns in the machine-part incidence matrix. The advantage of this approach is that both machine groups and part families can be obtained simultaneously. For an $M \times N$ matrix, there are $M!N!$ possible matrices obtained by permuting rows and columns. To reduce computational time, various heuristics have been used to find preferred matrix structure.

McCormick *et al.* (1972) developed the Bond Energy Algorithm (BEA) to form

the block diagonal matrix. BEA is a general clustering approach that attempts to maximize the bond energy between adjoining elements pairs. Rank Order Clustering (ROC) method is another rearranging approach proposed by King (1980). Later, King and Nakornchai (1982) revised the ROC method to overcome the computational difficulty and introduced ROC2 method. ROC and ROC2 are specifically designed for machine/part grouping, and consider the duplication of bottleneck machines. The deficiencies of the BEA and ROC methods were studied by Boe and Cheng (1991) who compared their close neighbour algorithm with BEA and ROC. To handle the problem of exceptional elements and machine duplication, Chan and Milner (1982) developed a Direct Clustering Algorithm (DCA).

To form the block diagonal matrices, Kusiak and Chow (1987a&b) proposed the Cluster Identification Algorithm (CIA).The CIA method was later used by Kusiak (1991) in his three branch and bound algorithms. A Hamiltonian path heuristic (HPH) proposed by Askin *et al.* (1991) is another rearranging approach that uses a distance measure based on dissimilarities of components. Chen and Irani (1993) developed the cluster first-sequence last heuristic by combining the clustering properties of the minimal spanning tree (MST) and the travelling salesman problem (TSP) to improve the process of permutation generation.

Simulated annealing

The simulated annealing (SA) algorithm, due to its ability to avoid local optima, has recently been employed to solve machine-part grouping problems. Liu and Wu (1993) tested the performance of an SA algorithm on the machine cell formation problem using two different cooling schedules, and concluded that their SA algorithm is able to find a near optimal solution within a polynomial time. Chen *et al.* (1995) also developed a

heuristic based on SA and showed that it performed well on a set of problems.

Methods based on operation sequence

The operation sequences have been accommodated in a number of studies in order to reduce material handling. Harhalakis *et al.* (1990) developed a two-phase heuristic capable of handling multiple non-consecutive operations. They used the operation sequence to define the intercell cost between each pair of cells. Then, in an iterative process, two cells with the largest normalized cost are aggregated together. A similar network approach was used by Wu and Salvendy (1993), but their algorithm attempts to find a minimal cut from a graph instead of considering the weight of arcs. Their network has a simple structure and can be extended to problems where the operation sequences are not available. Kern and Wei (1991) focused on reducing the number of intercell moves caused by exceptional elements. They proposed a method to eliminate exceptional elements by either duplicating of bottleneck machines or subcontracting the exceptional parts.

2.1.2 Mathematical programming

Mathematical programming is another approach that has been applied to machine cell formation problem. Kusiak (1985) developed a *p*-median model with the objective of maximizing the sum of similarities. Since a block diagonal matrix may not always be obtained, Kusiak (1987) extended his model to a generalized *p*-median model which can accommodate different process plans for each part. Thus, those process plans that yield a better diagonal matrix will be selected. Kusiak *et al.* (1986) further considered different versions of clustering problems where the number of cells and/or the cell size are given and fixed. The Lagrangian and eigenvector based methods were then used to solve their

models. The drawbacks of the *p*-median model have been extensively discussed by Srinivasan *et al.* (1990). These drawbacks are mainly due to the fact that in the *p*-median model, the number of cells is an input data rather than an outcome of the final solution. To alleviate this difficulty, Srinivasan *et al.* (1990) developed an assignment model for part grouping that uses a similarity matrix as the input. They compared their model with the *p*-median model and concluded that the assignment model is superior to the *p*-median model in terms of both solution quality and computational time.

Gunasingh and Lashkari (1989) formulated two 0-1 integer programming models to maximize the compatibility of machines and parts, which enables them to find a trade-off between the intercell moves and material handling.

More recently, Kusiak *et al.* (1993) considered two algorithms, one based on a quadratic programming model and the other a branch and bound algorithm, and compared them with ALC, BEA, ROC and ZODIAC. They concluded that the branch and bound algorithm performs better than the others. The grouping problem has also been formulated by Viswanathan (1995) as a quadratic integer programming. He then developed a heuristic to solve the problem.

A 0-1 linear integer programming model for the tool/part grouping problem in flexible manufacturing systems was formulated by Ventura *et al.* (1990). They developed a Lagrangian dual program to obtain an upper bound for the objective value. The dual program was then decomposed into a set of knapsack subproblems and solved using existing methods. A similar problem with fewer constraints was later solved by Cheng and Wu (1996).

2.1.3 Graph theory

Graph theory and vector methods have also been applied to the machine and part

grouping problem. Chandrasekharan and Rajagopalan (1986) developed an algorithm based on a bipartite graph to find an initial grouping. This initial solution was then improved by an ideal-seed method. The ideal-seed method was later improved and called ZODIAC (Chandrasekharan and Rajagopalan, 1987). In ZODIAC, the choice of seeds is not restricted, and a stopping criterion is used based on a relative efficiency. Vannelli and Kumar (1986) used graph theory to identify minimal bottleneck cells. A minimal bottleneck cell is the smallest set of machines and/or parts the deletion of which will result in a perfect block diagonal matrix. Similar approach were employed by Kumar and Vannelli (1987) to find the minimal number or minimal total cost of subcontractible parts.

Lee and Garcia-Diaz (1993) proposed a network flow approach to measure the dissimilarities between machine pairs, and to group them into the cells. They compared their network flow approach with the *p*-median model on a number of small to large size problems and showed that their approach outperforms the *p*-median model in both memory capacity and computational time.

Vector analytic method (VECAN) has been applied by Mukhopadhyay and Gopalakrishnan (1995) to the machine-part grouping problem. Their method consists of two major parts, i.e., machine grouping using vector analysis and group diagonalization.

2.1.4 Knowledge based systems

Studies along this line are scarce. Limited work was reported in the late 1980's. Kusiak and Heragu (1987) used an expert system to evaluate intermediate solutions of a problem with constraints. ElMaraghy and Gu (1988) proposed a knowledge based system to assign parts to existing machine cells. Their system uses a pattern recognition technique and a string of symbols representing required operations and part features.

2.1.5 Fuzzy set theory

A number of researchers have considered fuzzy set theory in machine-part grouping. Lee *et al.* (1991) used the fuzzy set concept to define a match index that indicates the degree of match between machines and parts. Then, they replaced the non-zero entries of the machine-part incidence matrix with the obtained degree of matches, and solved the problem using different approaches, namely ROC, BEA and a dual-objective simulated annealing approach. Their computations showed that simulated annealing approach was superior to the other two approaches.

A methodology for quantifying the data related to fuzzy features was proposed by Ben-Arieh and Triantaphyllou (1992). In their methodology, the discrete values of fuzzy features are compared to each other. Then, a relative weight is found for each feature, using pairwise comparison. Finally, parts are grouped according to the feature values and relative weights obtained.

Zhang and Wang (1992) incorporated the fuzziness into two traditional approaches. They first established a procedure to construct a non-binary machine-part matrix the elements of which are the degree of match between pairs of machines and parts. Then, they introduced a similarity coefficient based on the degree of match and used it in conjunction with SLCA and ROC.

Gindy *et al.* (1995) adopted the fuzzy C-means algorithm and added a new validity measure based on cluster compactness and machine repetition to achieve the minimum overlapping of cells (i.e., inter-cell moves) and maximum compactness of clusters (i.e., number of parts in a cell that require the full set of machines of that cell).

2.1.6 Neural networks

Another promising approach that has been widely applied to group technology in

recent years is based on neural networks. Neural networks are designed for parallel processing which is the core of the next generation of computers. Different topologies of neural networks have been introduced over the past two decades. Kao and Moon (1991) reported a back-propagation neural network approach for the part family formation problem, which has shown to be very efficient. The studies reported along this line include Kaparthi and Suresh (1991), and Moon and Chi (1992). However, the success of this approach depends on the training process which further relies on the availability of training data. As the machine cell/part family formation problem is a design problem, the training data are usually not available beforehand. The application of this method, therefore, could be very limited.

The binary adaptive resonance theory (ART1), originated from Carpenter and Grossberg (1986, 1988), has also been applied to machine grouping by several researchers (e.g., Kusiak and Chung 1991, Kaparthi and Suresh 1992, Liao and Chen 1993, Venugopal and Narendran 1994, Dagli and Huggahalli 1991, 1995). The advantage of ART1 is that it can handle large scale problems. However, this type of neural networks is sensitive to the order of input vectors. If the data related to the bottleneck machines are fed into the network first, a few very large machine groups will be resulted and thus the bottleneck machines have to be identified and removed before the grouping process. The identification and removal of the bottleneck machines are often fairly subjective. In addition, as pointed out by Chen and Cheng (1995), ART1 may fail to match some machines and cells with which they have a natural affinity. To avoid this problem, it has been suggested to reverse the 1's and 0's in the incidence matrix. However, a side effect of this remedy is that some machines may form a cell even if they do not have any similarity with each other (Chen and Cheng 1995).

Suresh and Kaparthi (1994) combined the fuzziness and ART1 and proposed an

algorithm using a choice parameter and a learning rate parameter. Similarly, Kamal and Burke (1996) proposed a fuzzy ART clustering algorithm (FACT) which is able to handle complex problems. Arizono *et al.* (1995) developed a stochastic neural network model based on the Hopfield network and applied it to part-tool grouping problem in FMSs. Their proposed system dynamics is similar to the Boltzmann machine, but the stochastic behaviour of the neuron output depends upon the sign of the net input instead of the value of the net input.

2.1.7 Pattern recognition technique

Though it is quite natural to group machines and parts using the pattern recognition technique, reported work in this field is very limited. Gu (1991) developed a modified Isodata algorithm and linked it with an optimization model that selects the parameter required by the Isodata algorithm. In the optimization model, the objective is to minimize the number of bottleneck machines while the parameters required by the Isodata algorithm are decision variables.

2.1.8 Performance measures for the binary grouping problem

The usefulness of the solution of the grouping problem largely depends on what performance measures are used. The early performance measures include the Bond Energy proposed by McCormick *et al.* (1972). however, this measure is highly dependent on the relative position of rows and columns. The same grouping solutions (in terms of grouping configuration) may have different bond energy values if the arrangements within cells are changed.

To avoid such dependence on the relative position of rows and columns, Chandrasekharan and Rajagopalan (1986) suggested a grouping efficiency (GE) which

has been widely used since then. The details of the grouping efficiency will be presented in Chapter 3. In a later study, Chandrasekharan and Rajagopalan (1989) analyzed the performance of the grouping efficiency in evaluating the solution qualities of a set of well-structured and ill-structured problems. The deficiency of the grouping efficiency was investigated by Kumar and Chandrasekharan (1990). However, this performance measure is still widely used in the literature.

The above literature review on the binary machine/part grouping indicates that the existing approaches suffer from at least one of the following shortcomings:

1) High computational efficiency and good solution quality cannot be simultaneously achieved.
2) Bottleneck machines and ill-structured problems cannot be handled.
3) They cannot escape local optima.
4) The solutions are highly dependent on the initial solutions.
5) Their performance depends on the order of the input data.
6) The number of cells has to be predefined.
7) Large sized problems cannot be solved.

An efficient approach free from all of these deficiencies cannot yet be found in the literature and thus needs to be developed.

2.2 Comprehensive Machine Cell/Part Family Formation

In a comprehensive grouping problem, some other manufacturing parameters such as processing times, expected demands and machine capacities are considered in order to increase the within-cell utilization and balance workload. To do so, a *timed* instead of

the binary machine-part matrix is used. This type of matrix requires processing times rather than the simple binary numbers as elements. The processing time can be expressed in time unit per piece or a fraction of machine. Related literature is reviewed in the following.

2.2.1 Heuristics

Ballakur and Steudel (1987) proposed a constructive heuristic for CMS design considering within-cell utilization, work load fraction, and cell size restriction. In their heuristic, each machine is admitted to a cell if its workload is higher than a predefined value. Those machines that do not satisfy the constraint are assigned to a remaining cell. A capacity limited, multiobjective heuristic for cell formation problem was suggested by Wei and Geither (1990). They also developed an enumeration scheme to obtain the optimal solution in order to validate the performance of the heuristic. Their objectives include the maximization of cell utilization, minimization of bottleneck cost, and minimization of intra and inter cell imbalances.

The machine capacity requirements (machine duplication) is the main concern of Sule (1991). He addressed the under-utilization problem when machines are duplicated and focused on material handling cost instead of intercell movements. He then developed a procedure based on this idea.

To minimize inter-machine flow, Okogbaa *et al.* (1992) developed a heuristic. In their approach, parts are assigned to machine groups after grouping machines based on the maximum number of within-cell operations. In case of a tie, the part is assigned to the least loaded cell. The least loaded cell is determined by processing times. They also conducted a simulation study and compared the system performance in three different layout patterns: process layout, original GT layout and alternative GT layout. The

alternative GT layout is essentially the same as the original, except that the workload is balanced by rerouteing parts to duplicated machines. They concluded that the alternative GT layout had higher performance than the other two.

Some emerging combinatorial search techniques have also been applied to the comprehensive machine grouping problem recently. Limited studies are briefly reviewed below.

Simulated annealing

The application of the simulated annealing method to the comprehensive grouping problem was carried out by Boctor (1996) who incorporated the operating and material handling costs into CMS design. He formulated an integer programming model which he then decomposed into a part assignment problem and a machine assignment problem. Since the machine assignment highly depends on part families, the first step (i.e., part assignment) is more important. A simulated annealing approach is accordingly developed to solve this problem.

Tabu search

Lot splitting combined with machine duplication has been considered by Logendran and Ramakrishna (1995) as a solution to capacity constrained problems. They studied a cell formation problem with three features. First, a machine can be duplicated in different cells if this leads to lower material handling cost; secondly, it is possible for a part to have two or more non-consecutive operations on the same machine; and finally, if the operation of a lot can not be fully processed by one machine because of its limited capacity, then that lot can be split into two. To solve this problem, Logendran and Ramakrishna (1995) first formulated the problem as a mathematical model with the

objective of minimizing the weighted sum of intercell and intracell moves. Then, they proposed a tabu search algorithm to find a near optimum solution.

Genetic algorithm

The only application of genetic algorithm to the grouping problem was done by Venugopal and Narendran (1992). In their study, a bicriteria genetic algorithm is developed. The objectives were to minimize the intercell moves and the total within cell load variation.

2.2.2 Mathematical programming

Co and Araar (1988) developed a three-stage procedure to a comprehensive machine grouping problem. In the first stage, operations are assigned to machines using a mathematical programming model with the objective of minimizing the deviation between workload and available capacity of each machine. Machine duplication is also considered in this stage. Then in the next stage, a rank order clustering algorithm (ROC) algorithm is used to arrange machines according to their similarities in terms of operations. Finally in the last stage, the size and composition of cells are obtained, using a direct search algorithm.

A multi-routeing problem with restricted machine capacities was studied by Nagi *et al.* (1990). Their approach considers capacity limitation in process plan selection. They formulated the problem as an NP-complete mathematical model. The problem was then decomposed into two subproblems. The first subproblem can be solved as a linear program, and the second subproblem is solved using heuristics.

Logendran (1991) used the processing time to accommodate the capacity constraint of each machine and focused on the identification of key machines in machine

grouping problems. He proposed three different methodologies to identify the key machine and developed a mathematical model to minimize the inter and intra cell moves. In another study, Logendran (1992) further considered machine duplication in the presence of operation sequence and budgetary constraint.

The uncertainty in product demand can also affect the CMS design. In view of this, Harhalakis *et al.* (1994) developed a mathematical model for a random product demand problem with the objective of minimizing the expected material handling cost and restriction on the cell size and machine capacities. They proposed a two-stage algorithm that first determines the mean values of the feasible production volumes and then provides a near optimal solution.

2.2.3 Knowledge based systems

The only work reported under this category is probably the study by Kusiak (1988). He considered a generalized formulation of the machine grouping problem with four constraints: 1) the available machining time on each machine, 2) the number of trips of material carriers to each cell, 3) the number of machines assigned to each cell, and 4) the mandatory inclusion of a machine in a cell due to technological requirements.

2.2.4 Neural networks

Neural networks have been applied to the comprehensive grouping problem in a number of studies. Rao and Gu (1995) proposed a neural network approach for machine cell formation problem considering machine duplication and capacity constraint. Their neural network uses binary input vectors and is similar to the binary adaptive resonance theory (ART1), but it has a different comparison phase. In their approach, processing times are used to calculate machine loads for an intermediate solution. In another study,

Rao and Gu (1994) combined the aforementioned neural network approach with an expert system. The expert system takes its input from the neural network and reassigns the exceptional parts, using alternative process plans.

The comprehensive cell formation problem was also studied by Suresh *et al.* (1995). They proposed a hierarchical procedure that includes three phases. The first phase consists of a neural network approach based on fuzzy ART1 to identify part families and associated machine types. In the second phase, a mixed integer goal programming model is solved to minimize new equipment cost and maximize cell independence. The third phase considers a 0-1 integer programming model that minimizes the inter-cell traffic. Both integer programming models in phases 2 and 3 are solved using the commercial software LINGO.

The literature review shows that a variety of manufacturing parameters have been included in CMS design. The most important parameters, processing times and machine capacities are commonly used. However, a grouping performance measure incorporating processing times has not yet been reported in the literature. Previous performance measures are often designed for binary machine/part grouping and thus cannot be used to evaluate the solution quality for the comprehensive grouping problems. A simulation study conducted by Seifoddini and Djassemi (1994) has shown that these performance measures are not consistent in predicting the performance of a cellular manufacturing system. They thus suggested that a performance measure based on workload be developed. In summary, the following limitations have been revealed in the reported studies on the comprehensive grouping problem:

1) The solutions are dependent on the initial solution.

2) There is a lack of complete compatibility between the primary CM goal (i.e.,

highly utilized independent cells) and the other objectives in the comprehensive grouping problem (such as maximizing the within-cell utilization and minimizing the workload imbalance).

3) Their performance depends on the order of the input data.

4) Large sized problems cannot be handled.

In view of the above, a generalized grouping efficiency index incorporating the processing times and lot sizes is developed in this thesis. This new measure will then be used to guide two algorithms (a simulated annealing algorithm and a neural network algorithm) to the grouping problem considering processing times, machine capacity limitations, lot sizes and capability of machine duplication.

2.3 Group Scheduling

As the GT concept helps reduce material handling and setups, group scheduling has attracted the attention of many researchers and practitioners. Some researchers focused on sequencing of parts and families on a single machine, while others considered the scheduling of parts and part families in a multi-machine cell. The related studies are briefly reviewed as follows.

2.3.1 Heuristics

Due to the complexity of the problem, heuristic approaches have been widely used for group scheduling. Vaithianathan and McRoberts (1982) investigated the decomposition of a part family into a set of subfamilies based on setup similarities. Then they proposed five heuristics for sequencing subfamilies and tested these heuristics against the SPT rule. It was concluded that this decomposition could yield a better

performance in terms of setup time savings and some other objectives. The decomposition of a part family into subfamilies was also considered by Solomon *et al.* (1995) whose objective was to minimize the total machine idle time in a flow shop cell. They formulated the problem as an NP-hard mathematical model. Four heuristics were then developed to solve this problem and the effectiveness of heuristics was examined through a simulation study.

The cyclic and acyclic group scheduling problems on a single machine with sequence-dependent setup times were addressed by Foo and Wager (1983). A cyclic scheduling problem arises when the production of the entire family is repeated periodically; if, instead of the entire family, only a set of selected parts from the family is to be re-produced, then the problem is called acyclic scheduling. Foo and Wager suggested that an acyclic problem could be converted to a cyclic problem by introducing a dummy "0" row and column to the setup time matrix, and then solved using the travelling salesman method.

Chan and Bedworth (1990) considered both static and dynamic scheduling methodologies for a flow shop cell. In a static cell scheduling problem, the job sequence is decided when a part family arrives at the cell. The job sequence in a dynamic case, however, is determined when the initial machine becomes available. Chan and Bedworth (1990) analytically studied this problem and proposed an algorithm for the static case. This method is then modified to solve the dynamic cell scheduling problem.

A cost based heuristic was developed by Kuo and Inman (1990) who formulated the group scheduling of a manufacturing cell as an economic lot scheduling problem (ELSP). In their approach, the average setup and inventory cost are minimized. Other heuristics for group scheduling problem are classified as follows.

Branch and bound method

This method was first used by Hitomi and Ham (1977) to minimize the total flow time in a flow shop cell. In their approach, family and job sequences are identical for all machines. Once the sequence of families and jobs are decided, job processing speeds are reduced to lower production cost. Ham *et al.* (1979) suggested a similar approach to minimize the total flow time with the minimum number of tardy jobs.

Simulated annealing

The only available reference on the application of simulated annealing to group scheduling is the study by Vakharia and Chang (1990). They compared SA with three other approaches in minimizing the makespan for a flow shop cell. They concluded that the simulated annealing heuristic performs better than the other heuristics, especially when the problem size increases.

Tabu search

The application of tabu search to group scheduling problems was reported by Skorin-Kapov and Vakharia (1993). They investigated the performance of a tabu search heuristic under different strategies to minimize the makespan in a flow shop cell and compared it with the simulated annealing heuristic proposed by Vakharia and Chang (1990). Their results indicate that for small problems, both approaches are comparable and for large problems, tabu search heuristic is preferred to the simulated annealing for the particular problem under their investigation.

2.3.2 Dynamic programming

Ozden *et al.* (1985) addressed the group scheduling problem as a Travelling

Salesman Problem (TSP) and pointed out that solving this problem by a dynamic programming approach can reduce computational time drastically. They showed that the optimal solution could be obtained for a problem of 30 jobs.

2.3.3 Simulation

The investigation of scheduling rules in cellular manufacturing is mostly carried out by simulation. Mosier *et al.* (1984) selected three GT scheduling rules in conjunction with five job shop rules and studied the effect of these rules on efficiency as well as effectiveness of a job shop cell.

The concepts of repetitive lots (RL) and truncated repetitive lots (TRL) were introduced by Flynn (1987). In an RL scheduling procedure, when the process of a job is completed on a machine, all waiting jobs will be scanned and a job identical to the job that just completed will be selected. In this way, the need for another setup will be eliminated. A TRL procedure is similar to RL, but the number of identical jobs that can be processed consecutively is limited. Flynn (1987) studied the effect of RL and TRL on the performance of a group technology shop, a traditional job shop and a hybrid GT-job shop when the setup times are sequence dependent. The simulation results indicate that RL procedures are beneficial in reducing setup times, queues length, waiting times, WIP inventory and flow time.

To maximize the throughput rate in a flow shop cell, Han and McGinnis (1988) developed a discrete time control (DTC) method. They compared the performance of DTC with another control method known as proportionally mixed flow (PMF) through a number of simulation experiments, and concluded that DTC performed better than PMF.

Some noteworthy research in simulation of group scheduling models has been

carried out by Mahmoodi *et al.* (1990a) to investigate dynamic group scheduling heuristics in a job shop cell. Three family selection rules (FCFAM, DDFAM and MSFAM) in conjunction with three part dispatching rules (FCFS, SPT and SI^x) under different experimental conditions have been examined. They used average job tardiness, average percentage of tardy jobs and average job time in system as performance measures. They then concluded that DDFAM and MSFAM were among the best family selection rules and SPT and SI^x dominated part dispatching rules. A similar study with different combination of family selection rules and part dispatching rules has also been carried out by Mahmoodi *et al.* (1992). The impact of several order releasing and due date assignment heuristics on CMS performance was also studied by Mahmoodi *et al.* (1990b).

The performance of several due date setting procedures in a flow shop cell was investigated by Russell and Philipoom (1991) through a simulation study. Wemmerlöv (1992) considered part family scheduling on a single machine and investigated the performance of four family dispatching rules under different experimental conditions. The effects of other factors such as setup to runtime ratio and cell load level on the performance of a selected set of group scheduling heuristics in a job shop cell were studied by Ruben *et al.* (1993).

Wirth *et al.* (1993) carried out a simulation study considering labour scheduling as well as family and part scheduling in a job shop cell. In their study, eleven decision rules were developed and tested under sixteen experimental conditions and four experimental factors. Their results indicate that reducing the variation of job arrivals and levelling the cell load can substantially improve the performance of the cell.

A recent study by Frazier (1996) indicates that most of the work in this area is based on simulation where the parts and part families are sequenced according to pre-

defined dispatching rules. The application of these rules are mostly limited to simple structured problems such as single machine sequencing and/or flow shop sequencing. Furthermore, when the group scheduling objective is related to due dates, the completion time of each operation becomes crucial. Since the completion time in a manufacturing system depends on machining speed, machining speed has to be included in the scheduling decision. However, this major factor has long been ignored in many studies. Only two studies considered the machining speed in group scheduling problem, i.e., Hitomi and Ham (1977), and Ham *et al.* (1979) but both considered a flow shop problem.

The above literature review clearly indicates the following important issues have been overlooked:

1) minimization of total weighted tardiness,
2) scheduling for job shop cell,
3) operation precedence and
4) machining speed.

As such, developing an efficient solution method for a group scheduling problem incorporating these issues is one of the main objectives of this thesis.

2.4 Motivation

The above literature review reveals that more effort can be devoted to CMS design and planning. For binary machine/part grouping, a fast and reliable approach that can efficiently handle large sized problem is not yet available. For the comprehensive machine/part grouping, attention has not been paid to incorporating the manufacturing

parameters into a generalized performance measure and thus the solutions are not relevant to today's manufacturing practice. Furthermore, there is no indication that the machining speed has ever been incorporated into the group scheduling model for a job shop cell. The machining speed becomes more critical when the objective of the model is related to due dates.

In view of the above, this thesis is directed toward the following:

1) Developing an efficient neural network approach for binary machine/part grouping that can handle bottleneck machines and ill-structured problems.
2) Developing a generalized grouping measure to incorporate processing times and lot sizes, and proposing: a) a simulated annealing search algorithm, and b) a neural network method to solve the problem.
3) Modelling the group scheduling problem with the objective of minimizing the total weighted tardiness that incorporates the machining speed and developing several alternative solution methods based on advanced search techniques.

CHAPTER 3

BINARY MACHINE CELL/PART FAMILY FORMATION

3.1 Background

One of the key issues in the CMS design is machine/part grouping. The task of machine/part grouping is usually carried out using a binary *machine-part incidence matrix*. Each nonzero entry in this matrix indicates the occurrence of processing a part on a machine. The data required to build this matrix are usually available in part flow sheets. With such matrices, machine/part groups can be identified by clustering the "1" elements. Figure 3.1 shows an example of a binary machine-part matrix and the associated grouping result.

91X

	Part						
Machine	1	2	3	4	5	6	7
1	1			1			1
2	1			1			1
3		1	1		1	1	
4	1			1			1
5		1	1		1	1	

(a) Machine-part incidence matrix

	Part						
Machine	2	3	5	6	1	4	7
3	1	1	1	1			
5	1	1	1	1			
1					1	1	1
2					1	1	1
4					1	1	1

(b) Grouping result

Figure 3.1. An ideal binary grouping problem

In reality, completely independent clusters (such as the solution shown in Figure 3.1) may not be obtained. It is likely to have some nonzero elements outside the clusters.

Such an element is called an *exceptional element* which represents an intercell movement. The performance of a grouping task is usually measured by the amount of intercell movements and the inside density of clusters. Figure 3.2 illustrates a grouping problem with several exceptional elements.

	Part						
Machine	1	2	3	4	5	6	7
1	1			1			1
2	1		1	1			1
3		1	1		1	1	
4	1						1
5	1	1	1	1	1		

(a) Machine-part incidence matrix

	Part						
Machine	2	3	5	6	1	4	7
3	1	1	1	1			
5	1	1	1		1	1	
1					1	1	1
2		1			1	1	1
4					1		1

(b) Grouping result

Figure 3.2. A grouping problem with exceptional elements

An exceptional element represents either a *bottleneck machine* or an *exceptional part*. In reality, a machine is considered a bottleneck if its workload is relatively higher than that of the majority of other machines. In the binary grouping problems where the processing times are assumed to be equal, the workload of each machine can be represented by the number of parts visiting that machine. Therefore, in this chapter, a machine is called a bottleneck machine if it is required by too many parts and it cannot be assigned to only one cell. A part is exceptional if it needs machines from more than one cell. In the problem shown in Figure 3.2, Machine 5 is a bottleneck machine and Parts 1,3 and 4 are exceptional parts. A good grouping method should be able to provide quick solutions to large problems and be able to deal with bottleneck machines as well as exceptional parts. As reviewed in Chapter 2, however, most existing methods cannot provide good solutions to large problems and the solution quality is often affected by the

presence of the exceptional elements as well as the machine-part matrix structure. In view of this, the main purpose of this chapter is to develop an efficient approach to large grouping problems and to overcome some difficulties present in existing methods.

3.2 Performance Measure

The *grouping efficiency* (GE) index proposed by Chandrasekharan and Rajagopalan (1986) is adopted in this study, though most of existing performance measures can be easily incorporated. The grouping efficiency is defined as follows:

$$\eta = q\eta_1 + (1-q)\eta_2 \tag{3.1}$$

where

$$\eta_1 = \frac{e_d}{\sum_{r=1}^{R} M_r N_r} \tag{3.2}$$

and

$$\eta_2 = 1 - \left(\frac{e_o}{mn - \sum_{r=1}^{R} M_r N_r} \right) \tag{3.3}$$

η_1 is a measure of the density of 1's in the diagonal blocks of the binary machine-part matrix. Since all operations are treated equally, η_1 can be considered as the cell utilization and therefore a higher value of η_1 is desired. η_2 represents the density of the 0's outside the diagonal blocks. The second term in Equation 3.3 measures the proportion of intercell moves. Therefore, a higher value of η_2 indicates a lower intercell movement level.

q is the weight ranging between 0 and 1. Many researchers have used q=0.5, but

Kumar and Chandrasekharan (1990) concluded that for any number of cells greater than 2, an equal value for q and $(1-q)$ may widen the disparity between the first and the second terms of Equation 3.1. They have suggested that q be selected based on the size and number of cells as follows:

$$q = \frac{\sum_{r=1}^{R} M_r N_r}{mn} \tag{3.4}$$

In the next section, the application of Hopfield neural network is discussed. Although there are different types of neural networks that may be applied to the grouping problem, the Hopfield network is the only one that specifically designed for optimization purpose. This neural network has been successfully applied to the Travelling Salesman Problem (TSP). Due to the similarity between the binary grouping problem and the TSP, the Hopfield network is adopted as a basis for a new algorithm.

3.3 Adaptation of Hopfield Neural Network

3.3.1 Hopfield neural network and travelling salesman problem

Hopfield (1982) introduced a network architecture known as the Hopfield network. His network has a single layer of neurons, and each neuron has a *state* that can be either binary (0,1) or bipolar (-1,1). In this thesis we use binary state. The entire network also has a state at any point in time. This state is represented by the states of a vector of neurons. The neurons in the Hopfield network are fully interconnected, i.e., each neuron is connected to every other neuron by a *synapse*, and every pair of neurons has connections in both directions.

Figure 3.3 shows a diagram of the Hopfield network. Because of this interconnection topology, the output of each neuron feeds into every other neuron and

causes the network to be recursive. This recursive feature will allow the network to maintain a stable state if no external input exists. Each interconnection has an associated *weight*. In the Hopfield network the connection weights of every pair of neurons have equal values in both directions, (i.e., $W_{xr,yq}=W_{yq,xr}$), and there is no self-loop connection, i.e., $W_{xr,xr}=0$. Connection weights have to be set prior to every application and maintained unchanged during updating procedures. These weight values are set in such a way that the *energy function* of the network is minimized.

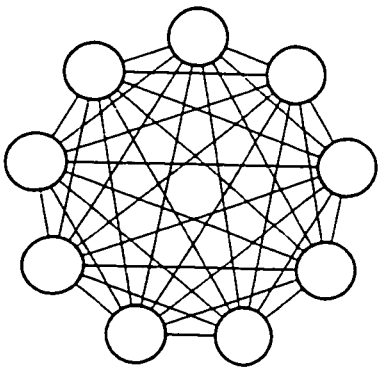


Figure 3.3. A Hopfield network

Before starting the updating procedure, an initial value has to be assigned to each neuron. Then one neuron at a time is updated and its output affects the state of other neurons. The updating candidate can be selected either randomly or sequentially. This updating process continues until the network stabilizes at a local or global minimum. The effect of each neuron on every other neuron is influenced by the weight of connection between them. Let U_{xr} denote the state of neuron xr (e.g., 0 or 1). This value will be the output of neuron xr to every other neuron which is a candidate for updating. If neuron yq is an updating candidate, it will be affected by all other neurons according to the following equation:

$$\tau_{yq} = \sum_{x=1}^{M} \sum_{r=1}^{M} U_{xr} W_{xr,yq} \qquad \forall y,q,\ xr \neq yq \tag{3.5}$$

The state of neuron yq will be updated as follows:

$$U_{yq} = \begin{cases} 1 & \text{if } \tau_{yq} \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad \forall\ y,q \tag{3.6}$$

The energy function in a Hopfield network is:

$$E = -\frac{1}{2} \sum_{x=1}^{M} \sum_{y=1}^{M} \sum_{r=1}^{M} \sum_{q=1}^{M} W_{xr,yq} U_{xr} U_{yq} \qquad xr \neq yq \tag{3.7}$$

Figure 3.4 illustrates the updating process in the Hopfield network. This process repeats itself until a stable state is attained. Based on this network topology, Hopfield and Tank (1985) proposed a model to solve the travelling salesman problem (TSP). In their model, neuron xr is active if the salesman visits city x in rth position of the tour and not active otherwise.

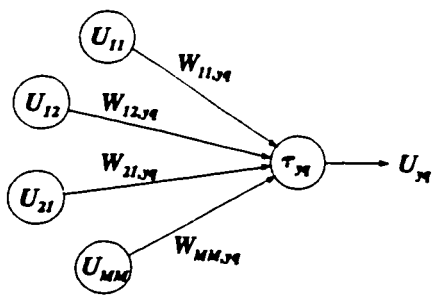


Figure 3.4. Updating process in Hopfield network

The energy function was selected in such a way that the energy level became lower after each state change. The progressive updating of the Hopfield network leads the network to a stable state at which the energy reached a minimum. This minimum could be either local or global. The following energy function was used in their network:

$$
\begin{aligned}
E = & \frac{A}{2} \sum_{x} \sum_{r} \sum_{\substack{q \\ q \neq r}} U_{xr} U_{xq} \\
& + \frac{B}{2} \sum_{r} \sum_{x} \sum_{\substack{y \\ y \neq x}} U_{xr} U_{yr} \\
& + \frac{C}{2} \left(\sum_{x} \sum_{r} U_{xr} - M \right)^2 \\
& + \frac{D}{2} \sum_{x} \sum_{\substack{y \\ y \neq x}} \sum_{r} l_{xy} U_{xr} \left(U_{y,r+1} + U_{y,r-1} \right)
\end{aligned}
\tag{3.8}
$$

The first term of the energy function corresponds to the constraint that each city should be visited only once. When this constraint is satisfied, this term has the minimum value equal to zero. The second term is associated with the restriction that the salesman can visit only one city at a time, i.e., no more than one city can be placed in the same position in the tour sequence. This term will be zero when each city is placed in one position. The third term will force the number of visited cities to be equal to M, i.e., all cities should be visited. Therefore, all the constraints in the TSP have been incorporated into the energy function. The objective of the TSP, minimizing total travel distance, is reflected by the last term.

It is important to note that, unlike most other neural networks, the Hopfield network does not need a training phase. Therefore, once the connection weights are set, they remain unchanged throughout the entire iterative process. The logic represented by the energy function is further mapped to the connection weights used in the updating process. Thus, identifying Equation 3.8 with Equation 3.7, the connection weights suggested by Hopfield and Tank (1985) have the following form:

$$
\begin{aligned}
W_{xr,yq} = & -A \, \delta_{xy} \left(1 - \delta_{rq} \right) \\
& - B \, \delta_{rq} \left(1 - \delta_{xy} \right) \\
& - C \\
& - D \, l_{xy} \left(\delta_{q,r+1} + \delta_{q,r-1} \right)
\end{aligned}
\qquad \forall \, x,y,r,q \tag{3.9}
$$

The first and second terms in the above weight equation are related to the constraints of the model (i.e., one city cannot be placed in more than one position in the tour, and one position cannot be used for more than one city). They therefore cause inhibitory connection within each row and each column. As A and B have the same effects but in different directions, in most cases A is set equal to B. The third term has a global inhibitory effect while the fourth term forces the network to minimize the distance between two adjacent cities in the tour.

It is worthwhile to note the relations between the corresponding terms in energy function and weight equation. Since these two equations might be different from one application to another, more discussions on the steps towards defining these equations are required. The energy function can be easily defined according to the objective and constraints of the problem under investigation. An objective of minimization will enter the energy function with a positive sign (e.g., the fourth term in Equation 3.8) and vice versa. The constraints are also added to the energy function with a positive sign in a way that violation from the constraint increases the value of the associated term. Once the objective function has been built, the weight equation will be accordingly defined. Every term in the energy function should have an associated term in the weight equation but with an opposite sign. These steps may be further clarified by examining the Equations 3.8 and 3.9 and also Equations 3.10 and 3.11 in the next section.

3.3.2 Application of Hopfield neural network to machine grouping

Considering the machines as cities and the cells as positions in a tour, the Hopfield network can be applied to the machine grouping problem. In this case, the constraints include: a) each machine can be assigned to only one cell, and b) all machines must be assigned to the cellular manufacturing system. Our objective is to maximize the

total similarity. By comparing the TSP and machine grouping problems, the following can be observed:

(1) In the TSP, each city can be visited only once. This is similar to the constraint of assigning one machine to only one cell in the machine grouping problem.

(2) In the TSP, no more than one city can be assigned to a single position, but in the machine grouping problem, it is allowed to assign more than one machine to a cell.

(3) The TSP requires that all the cities be visited, which is similar to the requirement that all machines must be assigned to the cellular manufacturing system in the machine grouping problem.

(4) The objective of TSP is to minimize the total travel distance while the objective of machine grouping problem is to maximize the similarity level.

The machine grouping problem can be expressed as the following mathematical model:

Maximize $$\sum_{r=1}^{R}\sum_{x=1}^{M-1}\sum_{y=x+1}^{M} S_{xy} U_{xr} U_{yr}$$

subject to:

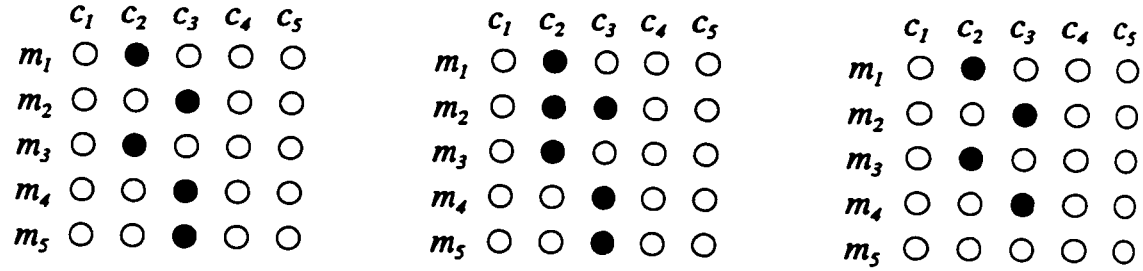
$$\sum_{r=1}^{R} U_{xr} = 1 \qquad \forall x$$

$$U_{xr} = 0 \text{ or } 1 \qquad \forall x, r$$

Note that in the above quadratic programming formulation, the number of cells R should be determined *a priori*, while in reality this number is unknown beforehand.

The machine grouping problem in the context of the Hopfield network can be illustrated in Figure 3.5. Figure 3.5(a) shows a feasible solution with Machines 1 and 3 being assigned to one cell and the remaining three machines to another. If machine

duplication is not allowed, the solution in Figure 3.5(b) is infeasible since Machine 2 is assigned to two columns, i.e., two cells. Figure 3.5(c) shows another infeasible solution where Row 5 is empty--no active neurons, i.e., Machine 5 is not assigned to any cell.



(a) A feasible solution

(b) An infeasible solution: m_2 is assigned to 2 cells

(c) An infeasible solution: m_5 is not assigned

Figure 3.5. Arrangement of neurons when applying the Hopfield neural network to a 5-machine grouping problem

(Note: m_x= machine x, c_r= cell r, ● active neuron, ○ inactive neuron)

Based on Observations (1) and (3), the first and the third terms in the TSP energy function are adopted for our problem. The second term is eliminated due to observation (2). The last term is modified to reflect the objective of the machine grouping problem. As a result, the energy function and the associated connection weight equations are as follows:

$$E = \frac{A}{2}\sum_{x}\sum_{r}\sum_{\substack{q \\ q\neq r}} U_{xr}U_{xq} + \frac{C}{2}\left(\sum_{x}\sum_{r} U_{xr} - M\right)^2 - \frac{D}{2}\sum_{x}\sum_{\substack{y \\ y\neq x}}\sum_{r} S_{xy}U_{xr}U_{yr} \tag{3.10}$$

Identifying Equation 3.10 with Equation 3.7, proper connection weights are:

$$w_{xr,yq} = -A\,\delta_{xy}(1-\delta_{rq}) - C + D\,S_{xy}\,\delta_{rq} \qquad \forall\, x,y,r,q \tag{3.11}$$

In this thesis, the Jaccard similarity coefficient is used, though any other meaningful similarity coefficient may be used in E and $w_{xr,yq}$. The Jaccard similarity coefficient is defined as follows (Anderberg 1973):

$$S_{xy} = \frac{a_{xy}}{a_{xy}+b_{xy}+c_{xy}} \qquad \forall\, x,y \tag{3.12}$$

The S_{xy} values can be easily obtained from the machine-part incidence matrix. As the 1's and 0's in the machine-part incidence matrix are only used to calculate the similarity coefficients which are fixed for a given grouping problem, the solution does not depend on the data input sequence. To initialize the grouping process, the number of cells is set to be equal to the number of machines and only one machine is assigned to each cell. The updating procedure mentioned above can be used to minimize the energy of the network, but the solution could be a local minimum. After reaching a stable state, the actual number of cells is equal to the number of cells that have at least one *active* neuron.

The grouping algorithm can be summarized as follows.

Algorithm 3.1

Step 1 (1) Read A, ΔA, C, ΔC, D, ΔD, M, G, S_{xy}, $w_{xr,yq}$, and set $G'=0$;

(2) Set $R \leftarrow M$, and initialize the network with one machine in each cell.

Step 2 Update network states until a stable state is attained.

Step 3 If any machine is assigned to more than one cell, discard the current solution, set $A \leftarrow A+\Delta A$, and go to (2) of Step 1. If the total number of active neurons (assigned machines) is less than M, discard the current solution, set $C \leftarrow C+\Delta C$, and go to (2) of Step 1.

Step 4 Update R and set $G' \leftarrow G'+R$. If $G'=G$, stop; else, go to the next step.

Step 5 Discard the current solution. If $G'<G$, set $D \leftarrow D-\Delta D$, otherwise, set $D \leftarrow D+\Delta D$. Go to (2) of Step 1.

The algorithm is coded in C and tested on a 486 PC.

3.3.3 Examples and implementation issues

The application of the proposed Hopfield neural network approach is illustrated using a 15-machine-10-part problem which has been analyzed by McAuley (1972), King (1980), and King and Nakornchai (1982). The associated machine-part incidence matrix is given in Figure 3.6(a). The final solution (Figure 3.6(b)) is obtained with less than 1 second of computational time. The solution is the same as that obtained using the single linkage clustering algorithm (McAuley 1972) and rank order clustering algorithm (King 1980, King and Nakornchai 1982).

It should be pointed out that the above example problem does not have any bottleneck machines. The updating process can converge to the final solution successfully. However, if any bottleneck machines are included, a feasible solution may not be obtained by applying the Hopfield model alone. Under such circumstances, any of the following cases is possible after reaching a stable state:

(a) Increasing A may fail to inhibit a machine from being assigned to more than one cell (i.e., for some rows, there exists more than one active neuron in each of them).

	Machine														
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Part															
01	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0
02	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1
03	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0
04	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0
05	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1
06	1	0	0	1	0	1	0	0	1	0	0	0	0	1	0
07	0	1	0	0	0	0	1	0	0	1	1	1	0	0	0
08	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1
09	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0
10	0	1	0	0	0	0	1	0	0	1	1	1	0	0	0

(a) Machine-part incidence matrix

	Cell														
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Machine															
01	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
02	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
03	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
04	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
05	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
06	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
07	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
08	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
09	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
13	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
15	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

(b) Final grouping result; actual number of cells: 3
Cell 1: 3, 5, 8, 13, 15
Cell 2: 1, 4, 6, 9, 14
Cell 3: 2, 7, 10, 11, 12

Figure 3.6. A grouping problem without bottleneck machines

(b) Increasing C may fail to prevent some machines from being unassigned (i.e., some rows may not have any active neuron).

(c) The desired number of machine cells may not be formed by decreasing D alone.

It is interesting to note that case (a) may indicate that the machine being assigned to more than one cell is very likely a bottleneck machine and if so, duplication may be considered. However, if there is no extra machine of that type for duplication, the bottleneck machine cannot be duplicated. Then further computation is required to assign it to only the cell that needs it most, i.e., the cell has the highest priority for "owning" such a machine. Case (b) may reveal the fact that the unassigned machines are simply not compatible with any of the existing cells and therefore, it may be desirable for them to form a separate cell. Therefore, the following external manipulations are adopted:

(1) If case (a) arises, it indicates that a potential bottleneck machine may exist. Then the machine cells that do not have such potential machines will be removed and the program is re-run only for the cells that contain the potential bottleneck machines. If the new run still cannot allocate the potential bottleneck machine to a single cell, it indicates that the machine is a "true" bottleneck machine and should be duplicated. If duplication is not allowed, the machine will be assigned to a cell to which it has a higher similarity relationship. In our computations, most such potential bottleneck machines can be assigned to a single cell after a limited number of runs.

(2) In case (b), if there are many unassigned machines, machines that have been assigned to cells are removed and the program is re-run for only the unassigned machines. This generally can separate the machines into several cells. If there are only a few, say, 2 or 3 machines unassigned, they are

simply assigned to an existing cell starting with the smallest existing cell and the program is run again. If they can be allocated to an existing cell after re-running the program, the process stops; otherwise, they form a separate cell.

Accordingly, Algorithm 3.1 is modified to accommodate the above external manipulations. The modified algorithm is given below.

Algorithm 3.2

Step 1 (1) Read Γ, Φ, ϵ, Ψ, A, ΔA, C, ΔC, D, ΔD, M, G, S_{xy}, $w_{xr,yq}$, and set $G'=0$, $\Lambda=\{\phi\}$, $\rho=\{\phi\}$.

(2) Initialize the network with one machine in each cell.

Step 2 Update network states until a stable state is attained; update Λ, ρ, Ψ, G'.

Step 3 IF $\Lambda\neq\{\phi\}$, set $A\leftarrow A+\Delta A$.

IF $A<MA$, discard the current solution and go to (2) of Step 1.

ELSE set $\Phi=\Gamma-\rho$, record the successful cells, and go to (2) of Step 1.

ENDIF

ELSE go to the next step.

ENDIF

Step 4 IF $\Psi\neq\{\phi\}$, set $C\leftarrow C+\Delta C$.

IF $C<MC$, discard the current solution and go to (2) of Step 1.

ELSE IF the number of machines in Ψ is greater than ϵ, set $\Phi\leftarrow\Psi$, record the successful cells, and go to (2) of Step 1.

ELSE set $\Phi=(\Psi \vee$ an existing cell), record the other successful cells, and go to (2) of Step 1.

ENDIF

ENDIF

ELSE go to the next step.

ENDIF

Step 5 Update G'

IF $G'=G$, stop

ELSE, go to the next step.

Step 6 Set $D \leftarrow D+\Delta D$.

IF $D<MD$, discard the current solution and go to (2) of Step 1

ELSE go to the next step.

Step 7 Order the current machine cells such that

$M_1 \geq M_2 \geq \ldots \geq M_r \geq \ldots \geq M_R$

set $\Phi \leftarrow$ {set of machines in the first cell} and go to (2) of Step 1.

In this algorithm, a successful cell is one that does not contain the machines in sets Λ or Ψ.

The application of Algorithm 3.2 is illustrated using a grouping problem initially investigated by Burbidge (1975). The machine-part matrix is shown in Figure 3.7. The desired number of cells G is set to 5. Some intermediate sample solutions and the final solution are listed in Table 3.1. The first sample solution consists of two cells. Machines 6 and 8 are assigned to both cells. Then the algorithm breaks the two cells into five as shown in the second sample solution. Again both Machines 6 and 8 are assigned to more than one cell, indicating that they are potential bottleneck machines. The final solution includes five cells and Machines 6 and 8 are assigned to a single cell, Cell 2. It took about 2 seconds to get the final solution on a 486 PC using a C program developed based on Algorithm 3.2.

	Machine															
Part	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
01	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0
02	0	1	0	0	0	1	0	1	1	0	0	0	0	1	0	1
03	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0
04	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
05	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0
06	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
07	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1
08	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
09	0	0	0	1	1	0	0	1	0	0	1	0	0	0	0	0
10	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
12	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
13	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0
14	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0
15	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
16	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
17	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0
18	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
19	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1	0
20	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
21	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0
22	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
23	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0
25	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
27	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0
28	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0
29	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
31	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
32	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	1
33	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
34	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
35	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
36	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
37	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	1
38	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1
39	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
40	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
41	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0
42	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	1
43	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0

Figure 3.7. Machine-part incidence matrix of Burbidge's problem

Sample solutions	Machine cells	Part families	Λ
1	{1,2,3,4,5,6,8,9,14,15,16}	{6,7,8,10,11,12,13}	{6,8}
2	{1,2,6,8,9,16} {4,5,6,8,15} {3,6,8,14}	{6,7,8,10} {8,11,12,13}	{6,8}
final	{1,2,9,16} {4,5,6,8,15} {3,14}	{7,10} (11,12,13}	{ϕ}

Table 3.1. Intermediate and final solutions of the example problem shown in Figure 3.7

Some insights can be observed by examining both the intermediate and final solutions. For instance, sample Solution 2 suggests that if additional machines are available, Machines 6 and 8 may be duplicated. However, if additional machines are not available or machine duplications are not allowed, Machines 6 and 8 should be assigned to the second cell as shown in the final solution. The reason is that the total similarity level can be increased.

It is noted that the obtained solution will be identical to that of Kaparthi and Suresh (1992) if Machines 6 and 8 are removed from the second cell in the final solution. Further, by combining the first and the third cells of sample Solution 2, a solution identical to the one reported by King and Nakornchai (1982) can be obtained.

In our computations, it is found that the selection of the values of the tunable parameters, A, C and D is important to obtain a feasible solution. It is also observed that the final solution is not very sensitive to the variations of these parameters, i.e., the same solution can be obtained when the parameters are changed in relatively small ranges. The guidelines used in selecting these parameters are:

(1) If a machine is assigned to more than one cell, increase A.

(2) If the total number of active neurons (assigned machines) is less than M, increase C.

(3) If the actual number of cells is greater than the specified, increase D.

Further, it should be emphasized that by using the above approach a) the lengthy training process is not required; b) the solution is insensitive to the sequence of the input data (vectors) and the tedious trial process, needed in Carpenter-Grossberg method, is avoided in searching a more preferred solution; and c) the identification and removal of bottleneck machines can be automatically done by the Hopfield network based algorithm which otherwise have to be done manually.

Nevertheless, the computations have revealed that large cells might result when bottleneck machines exist, and such cells may be decomposed into smaller cells by some trial-and-error external manipulations. Another potential difficulty associated with the original Hopfield network is how to initialize the network since it is desirable to start from a feasible state. So far, the network is initialized by assigning a single machine to each cell. However, different initial network states may result in quite different solutions. In general, there are $M!$ feasible initial states. It might be computationally prohibitive to enumerate all the initial states when the problem size is large. In summary, issues remaining to be addressed in applying the Hopfield network to machine grouping include: a) computational time, b) local optima, c) large cell solutions, and d) initial-state-dependent solution. To this end, a new topology of Hopfield network and an objective-guided search scheme are proposed in the following section.

3.4 Ortho-Synapse Hopfield Network (OSHN)

3.4.1 OSHN structure

Further examining the structure of the Hopfield neural network and the notion of machine grouping indicates that only the horizontal and vertical connections are relevant to the machine grouping problem. This can be illustrated jointly by Figure 3.8 and the

energy function (Equation 3.10). The second term of the energy function requires that the network should have exactly M active neurons. The M active neurons should be distributed in such a way that

1) each row has only one active neuron (enforced by the first term of the energy function); and
2) multiple active neurons may be assigned to a single column if the overall similarity can be improved (reflected by the third term of the energy function).

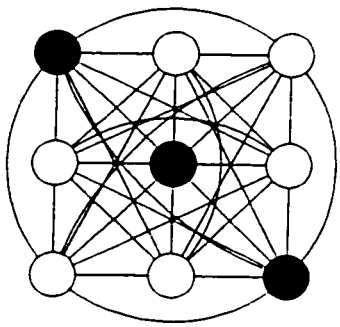
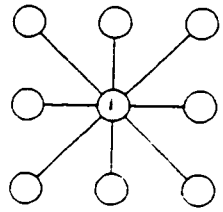
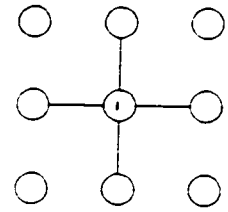


Figure 3.8. Connections in the original Hopfield network

Therefore, both the constraints and the objective of the machine grouping problem have been accommodated by the horizontal and vertical connections in the Hopfield neural network. The *oblique* connections in the network seem to be unnecessary. For this reason, the oblique connections are removed and thus a simpler Hopfield network structure is obtained (Figures 3.9 and 3.10). The modified neural network is called *Ortho-Synapse Hopfield Network* (OSHN) as it contains only horizontal and vertical synapses.

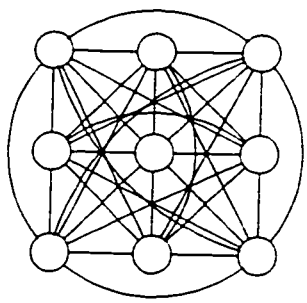


(a) Original Hopfield network

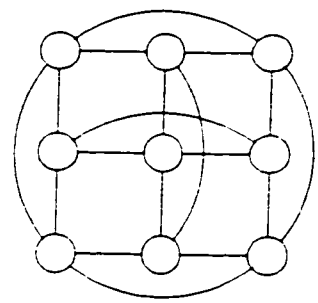


(b) OSHN

Figure 3.9. Connections between Neuron *j* and other neurons (for a 3-machine problem)



(a)



(b)

Figure 3.10. Connections among all neurons: (a) original Hopfield network, (b) OSHN (for a 3-machine problem)

The energy function of the OSHN is the same as Equation 3.10. However, due to the new topology of the OSHN, the weight equation shown in Equation 3.11 is changed to:

$$\omega_{xr,yq} = \begin{cases} w_{xr,yq} & \text{if } (x=y \ \& \ r\neq q) \text{ or } (x\neq y \ \& \ r=q) \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, x,y,r,q \qquad (3.13)$$

where $w_{xr,yq}$ is the weight equation defined in Equation 3.11.

The state updating threshold τ_{yq} is obtained by substituting $\omega_{xr,yq}$ into Equation 3.5:

$$\tau_{yq} = \sum_{x=1}^{M} \sum_{r=1}^{M} U_{xr}\, \omega_{xr,yq} \qquad \forall\, y,q,\ yq \neq xr \tag{3.14}$$

The number of connections is considerably reduced by removing the oblique synapses. If a bidirectional connection between two neurons is treated as a single synapse, the original Hopfield network for an M-machine problem will have $M^2(M^2\text{-}1)/2$ synapses, as compared to only $M^2(M\text{-}1)$ synapses in an OSHN. As the major portion of the computational time is spent in state updating of the neural network, the fewer synapses are used, the less computational time will be required. Thus, much shorter computational time can be expected by using the OSHN.

3.4.2 Application of the OSHN to machine grouping

Given the above neural network structure and equations, an OSHN based algorithm for machine grouping can be outlined as follows:

Algorithm 3.3

Step 1 Select A, C, D and compute S_{xy}.

Step 2 Calculate $\omega_{xr,yq}$ using Equation 3.13.

Step 3 Update the network using Equations 3.14 and 3.6.

Step 4 If the state of the network is the same as that of the last iteration, stop; otherwise, go to Step 3.

An illustrative example

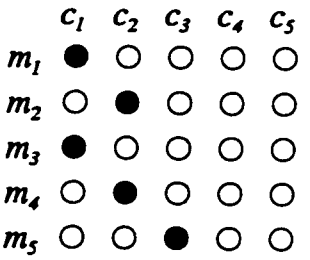
Consider a 5-machine-6-part problem. The machine-part incidence matrix for this problem is shown in Figure 3.11.

Machine	Part					
	1	2	3	4	5	6
1	1			1	1	
2		1	1			1
3	1			1	1	
4		1	1			1
5	1	1	1	1	1	

Figure 3.11. The machine-part incidence matrix

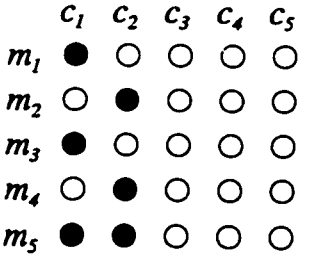
The problem contains one bottleneck machine, i.e. Machine 5, which is required by five parts. The parameters (A, C, D) are set to (0,1,1) and the problem is solved using Algorithm 3.3. The solution is shown in Figure 3.12(a). As can be seen in Figure 3.12(a), three cells, corresponding to the three columns containing at least one active neuron, are formed. Cell 1 includes Machines 1 and 3, Cell 2 contains Machines 2 and 4, and the last cell contains only Machine 5. It is noted that Machine 5 is not an unassigned machine since the associated row is not empty.

As stated above, a preferred solution may not be obtained using a particular set of parameters. Each parameter, A, C or D has a specific effect on the final solution. Parameter A is related to the constraint of "assigning each machine to only one cell". Therefore, a lower A may result in the duplication of bottleneck machines. Parameter C is used to ensure that all machines have been assigned. Parameter D is the weight of the objective function of the grouping problem. If properly selected, D can force similar machines to form cells so that the overall similarity level can be improved. Therefore, parameters A, C, and D can be tuned to obtain preferred solutions.



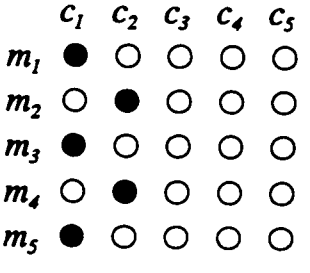
Machine	Part 1	4	5	2	3	6
1	1	1	1			
3	1	1	1			
2				1	1	1
4				1	1	1
5	1	1	1	1	1	

(a) Final neuron states and solution using parameters (0,1,1)



Machine	Part 1	4	5	2	3	6
1	1	1	1			
3	1	1	1			
5	1	1	1			
5				1	1	
2				1	1	1
4				1	1	1

(b) Final neuron states and solution using parameters (0,1,5)



Machine	Part 1	4	5	2	3	6
1	1	1	1			
3	1	1	1			
5	1	1	1	1	1	
2				1	1	1
4				1	1	1

(c) Final neuron states and solution using parameters (1,1,5)

Figure 3.12. Solutions obtained using different parameters for the example problem shown in Figure 3.11

For instance, in the above example it may not be desirable to have a single-machine cell. If so, Machine 5 has to be either allocated to one of the two existing cells or duplicated for both. This can be done by adjusting parameter D. For instance, if parameters are changed to (0,1,5), a new solution is obtained as shown in Figure 3.12(b). In this case, Machine 5 is assigned to two cells. If machine duplication is allowed, this may be a reasonable solution. However, if an extra machine 5 is not available or machine duplication is not allowed, Machine 5 has to be allocated to only one of the existing cells. Since parameter A is used to prevent a machine from being assigned to more than one cell, A can be increased. By changing the parameters to (1,1,5), machine duplication is avoided and Machine 5 is allocated to only cell 1. The solution is shown in Figure 3.12(c).

Many trial problems have been solved to obtain further insights into the tunable parameters. Our computational experience has shown that parameters C and D are related to each other in such a way that D should be at least equal to C. Otherwise, very few large cells or too many small cells may result. Both C and D influence the final number of machine cells. A lower value of C leads to fewer cells while a higher value of C intends to yield more cells. D affects the number of cells in an opposite direction. However, the final number of cells is more sensitive to the change of C. For all the trial problems, machine duplications can be prevented by increasing A and small cells can be combined to existing cells by increasing D. Based on the above observations, the following guidelines are suggested in parameter tuning:

(1) If undesirable machine duplications are observed, increase A.

(2) Increase C if there are too few cells and decrease C when there are too many cells.

(3) If too many small cells are formed, increase D.

Although many undesired solutions can be avoided by tuning A, C and D, the following cases may arise:

(a) Increasing C may fail to prevent some machines from being unassigned (i.e., some rows in the network may be empty--no active neurons), and

(b) the final number of cells obtained from the neural network solution may be greater than the maximum allowed number of cells and it cannot be further adjusted by changing C and D.

To handle the above cases, the following is proposed:

If case (a) is observed, the *average similarity*, $\hat{S}_{yq}$, for every unassigned machine is calculated:

$$\hat{S}_{yq} = \frac{\sum_{x=1}^{M} S_{xy} U_{xq}}{\sum_{x=1}^{M} U_{xq}} \qquad \forall\, q,\ \ y \in \Psi \tag{3.15}$$

An unassigned machine y will be allocated to a cell r if $\hat{S}_{yr}=max\{\hat{S}_{yq} | q=1,\ldots,R\}$.

If case (b) occurs and the maximum number of allowed cells is K, an $M\times K$ network instead of the original $M\times M$ network will be used. Accordingly, the M in the second summation of Equation 3.14 is replaced by K and thus this restriction can be satisfied.

Now, an algorithm can be developed based on the above parameter tuning guidelines and discussions. However, the energy level may not be directly used to measure the "goodness" of a solution. For instance, suppose two energy levels, E_a and E_b, $E_a > E_b$, can be calculated based on two different solutions obtained using two sets of parameters. It appears that the solution associated to E_b is preferred as lower energy level is achieved. However, this does not necessarily mean that its associated grouping

efficiency level, η_b is higher than η_a. It is therefore difficult to compare the quality of the solutions obtained using different parameters based on the energy level. Hence, we suggest that the GT performance measure discussed in Section 3.1 be used instead of the energy level to guide the tuning process. The details are discussed in the next section.

3.4.3 An objective-guided algorithm

In this section, an objective-guided search approach is presented. The basic idea is that the tuning or search direction should be guided by the objective value (η, in our case). The search starts from an initial set of (A,C,D) and the η is calculated at the end of the iteration. Then a new iteration is performed with a set of adjusted parameters and the new η is calculated. If the new η is higher (better) than the previous one, the parameters will be adjusted in the same "direction" (increase or decrease) for the next iteration. Otherwise, stop. To further reduce the chance of being trapped into a local optimum, we propose to search a few additional steps in the same direction. If all the steps yield worse solutions or the same solution, stop. Otherwise, a better solution is obtained and the search will continue. The procedure is repeated until no objective value improvement can be achieved.

To implement the objective-guided approach, the performance measure η has to be updated after each iteration. To calculate η, both part families and machine groups have to be identified. The information about machine groups can be obtained from the neural network. The part family information, however, is not readily available from the network output. In order to identify part families, we introduce a membership measure, D_{rl}, which reflects the belongingness of part l to cell r. D_{rl} is defined as follows:

$$D_{rl} = \frac{m_{rl}}{M_r} \qquad \forall r,l \tag{3.16}$$

where m_{rl} and M_r can be easily obtained from the machine-part matrix and current cell configuration. Then part l will be assigned to cell q if $D_{ql}=max\{D_{rl}|r=1,\ldots,R\}$. D_{rl} ranges between 0 and 1. $D_{rl}=0$ means that none of the machines in cell r is required by part l, and $D_{rl}=1$ indicates that all machines of cell r are required by part l. The advantage of using the above membership function is that it reflects the machine utilization in each cell by each part. For example, $D_{rl}=2/3$ shows that, if part l is assigned to cell r, 2/3 of the machines in cell r will be utilized by part l. Now, every part can be assigned to a particular machine cell and all the parts assigned to a machine cell are considered as a part family. The performance measure η can then be calculated.

The objective-guided algorithm is presented below.

Algorithm 3.4

Step 1 Set $A\leftarrow 0$, $C\leftarrow 1$, $D\leftarrow 1$, $\eta\leftarrow 0$, $CTR\leftarrow 0$, read ΔA, ΔC, ΔD, u, NS and compute S_{xy}.

Step 2 If the maximum number of cells is pre-specified as K, set $V\leftarrow K$; otherwise, set $V\leftarrow M$.

Step 3 Calculate $\omega_{xr,yq}$ using Equation 3.13.

Step 4 Update the network using the following equation and Equation 3.6

$$\tau_{yq} = \sum_{x=1}^{M}\sum_{r=1}^{V} U_{xr}\,\omega_{xr,yq} \qquad \forall y,q,\ xr\neq yq \tag{3.17}$$

Step 5 If the state of network is the same as that of the last iteration, go to the next step; otherwise, go to Step 4.

Step 6 If there is any unassigned machine, say machine y, find $\hat{S}_{yr}=max\{\hat{S}_{yq}|q=1,\ldots,R\}$ and assign machine y to cell r; otherwise, go to the

next step.

Step 7 Identify part families based on the membership measure defined in Equation 3.16.

Step 8 If duplicated machine assignments are observed and they are not allowed, set $A \leftarrow A + \Delta A$ and go to Step 4; otherwise, go to the next step.

Step 9 If the number of small cells is greater than u, set $D \leftarrow D + \Delta D$ and go to Step 4; otherwise, go to the next step.

Step 10 If all machines have been grouped into a single cell, set $C \leftarrow C + \Delta C$ and go to Step 4; otherwise, go to the next step.

Step 11 Calculate the grouping efficiency, φ, obtained in the current iteration. If $\varphi > \eta$, set $\eta \leftarrow \varphi$, $D \leftarrow D + \Delta D$, $CTR \leftarrow 0$ and go to Step 4; otherwise, $CTR \leftarrow CTR + 1$, and go to the next step.

Step 12 If $CTR < NS$, go to Step 4; otherwise, discard the current solution and accept the solution corresponding to η as the final solution.

In this algorithm, u is the maximum number of allowed small cells, and NS is the allowed number of additional search steps in the same direction when a worse solution is detected. The size of "small cell", values of u and NS are specified by the decision maker.

3.4.4 Computational results and discussion

3.4.4.1 Results

The OSHN algorithm has been coded in C, and run on a 486 PC with a 33 MHz processor and 4 MB RAM. Twenty eight notable problems have been selected from the literature to test the performance of the algorithm. The solutions of the 28 problems are

cited from Chen and Cheng (1995). Alternative solutions can be obtained by using pre-determined maximum number of cells or leaving the final number of cells unrestricted. We provide two solutions for each of the 28 problems. The first solution is solved without restricting the number of cells. The second solution is obtained with the number of cells being set to be equal to that of the reference. The computational results are summarized in Table 3.2. For comparison purpose, we have also listed the solutions obtained using non-neural algorithms, basic ART1 and modified ART1 algorithms (Chen and Cheng 1995). The modified ART1 is an improved ART1 proposed by Chen and Cheng (1995).

Table 3.2 shows that, as compared to the non-neural algorithms, the objective-guided OSHN approach (with restricted number of cells) provides higher η values for 6 problems (2, 12, 14, 16, 27, 28), slightly lower η value (79.62% vs. 79.86%) in only one case (problem 3), and identical solutions for the remaining 21 problems. It is shown in Table 3.2, 9 out of the 28 problems cannot be solved by using the basic ART1 alone while the OSHN method can provide solutions for all the 28 problems. The OSHN (with restricted number of cells) solutions are identical to those of the basic ART1 for the 19 problems that can be solved by the basic ART1. In comparison with the modified ART1, the OSHN method (with restricted number of cells) provides higher η values for problems 3, 12, 27, and 28, lower η values for problems 2 and 14, and identical solutions for the rest. If the number of machine cells is not restricted, the OSHN algorithm will yield 9 higher η solutions and 19 identical solutions as compared with the modified ART1. Different η values are caused by different machine or part compositions. For example, for problem 3, the machine cell configuration obtained using our algorithm is the same as that of Chen and Cheng (1995) but the part compositions are different. The part families in (Chen and Cheng 1995) are:

PF-1: (2,4,10,18,28,32,37,38,40,42)

PF-2: (6,7,17,34,35,36)

PF-3: (5,8,9,12,14,15,16,19,21,23,29,33,41,43)

PF-4: (1,13,25,26,31,39)

PF-5: (3,11,20,22,24,27,30)

The part families obtained from the OSHN algorithm are:

PF-1: (2,4,10,18,28,32,37,38,40,42)

PF-2: (6,7,17,34,35,36)

PF-3: (5,8,9,14,15,16,19,21,23,29,33,41,43)

PF-4: (1,12,13,25,26,31,39)

PF-5: (3,11,20,22,24,27,30)

3.4.4.2 Discussion

(1) Computational times are reasonably short

The computational times are reasonably short in both cases. For the first set of solutions, (i.e., the number of cells is not restricted), the computational times are longer since the search starts with larger networks. Some problems need considerably longer time, e.g., 37.75 seconds for problem 8 and 24.07 seconds for problem 14. This is, in our opinion, due to the problem size and complexity. Problem 8 has 40 machines and 100 parts. If the number of machine cells is unrestricted, the number of cells will be assumed to be equal to the number of machines and hence the search will begin with a large network. However, since the number of machine cells is considerably less than the number of machines in practice, the computational time can be substantially reduced by specifying a preferred number of cells. This has been demonstrated by our second set of solutions. For instance, the computational time for problem 8 (40 machines and 100

parts) is only 4.18 seconds if the number of cells is limited to 10.

The problem complexity may be affected by the number of exceptional elements in the machine-part incidence matrix. For example, problems 9, 10, and 11 have the same number of parts and machines. Problem 9 has no exceptional element while problems 9 and 10 have respectively 10 and 20 exceptional elements after grouping (Chandrasekharan and Rajagopalan 1989). The computational times for the three problems (3.74 seconds, 6.15 seconds and 12.80 seconds respectively) have, to certain extent, reflected the complexity of the problems.

To further demonstrate the computational efficiency of the proposed algorithm, a larger problem with 50 machines and 150 parts has also been solved. The original machine-part incidence matrix is shown in Figure 3.13(a) and the number of machine cells is specified as 6. It took only 3.13 seconds to obtain the solution (Figure 3.13(b)) with grouping efficiency $\eta = 81.9\%$.

(2) Local optima problem is alleviated

Local optima problem is the inherent nature of the Hopfield neural network (Dayhoff 1990). The network may stabilize at quite different energy levels -- very likely, local optima -- when different sets of parameters are used. The objective-guided search approach will lead the search from one local optimum to another towards a higher objective value by tuning the parameters in a favourable direction. This has been demonstrated by the comparisons shown in Table 3.2. Most of the solutions from the literature are at least near optimum. Our solutions are favourably comparable with these solutions. The local optima problem is thus eased.

(3) Large cell problem has been avoided

The large cells due to the bottleneck machines have forced researchers to perform some external manipulations which in many cases are subjective. With the OSHN based algorithm, this problem has been automatically solved by assigning the bottleneck machines to the proper cells if machine duplications are not allowed. This can be clearly demonstrated by the example problem shown in Figure 3.12. The two bottleneck machines (machines 30 and 50) have been automatically assigned to the cells to which they have the highest belongingness without causing large cells.

(4) Initial-state-dependency has been reduced

Unlike in the original Hopfield neural network, the solutions of the OSHN are not sensitive to the initial states of the neural network. In fact, we have performed all the computations starting with fully inactive networks, i.e., all the neurons are inactive, which are obviously infeasible, and all the solutions quickly converge to reasonably good results.

It should be pointed out that, for some test problems, e.g., problem 14, the number of cells obtained in the first set of solutions may be considerably greater than the one from the literature. This is mainly caused by the performance measure. If a different performance measure is used, the number of cells may be different. Therefore, a different performance measure may be used to "guide" the search process. The selection of the performance measure, however, depends on the user's preference.

No.	Problem Source	Non-neural algorithm reference[a]	Size ($m \times n$)	Reference solutions: Non-neural g	Reference solutions: Non-neural η (%)	Reference solutions: Basic ART1[b,c] g	Reference solutions: Basic ART1[b,c] η (%)	Reference solutions: Modified ART1[b] g	Reference solutions: Modified ART1[b] η (%)	OSHN: unrestricted number of cells g	OSHN: unrestricted number of cells η (%)	OSHN: unrestricted number of cells Time[d]	OSHN: restricted number of cells g	OSHN: restricted number of cells η (%)	OSHN: restricted number of cells Time[d]
1	Askin *et al.* (1991)	--	10×15	3	91.50	3	91.50	3	91.50	3	91.50	0.27	3	91.50	0.05
2	Boe and Cheng (1991)	--	20×35	4	77.36	*	*	4	80.38	5	81.38	7.69	4	79.79	0.55
3	Burbidge (1975)	Askin *et al.* (1991)	16×43	5	79.86	*	*	5	79.40	5	79.62	2.97	5	79.62	0.55
4	Burbidge (1975) (excluding machines 6 and 8)	Askin *et al.* (1991)	14×43	5	83.02	5	83.02	5	83.02	5	83.02	1.48	5	83.02	0.33
5	Burbidge (1975) (duplicating machines 6 and 8)	Kusiak and Cho (1992)	22×43	5	80.70	5	80.70	5	80.70	5	80.70	5.88	5	80.70	0.66
6	Carrie (1973)	--	20×35	4	87.81	4	87.81	4	87.81	4	87.81	4.84	4	87.81	0.44
7	Chandrasekharan and Rajagopalan (1986)	--	8×20	3	95.83	3	95.83	3	95.83	3	95.83	0.11	3	95.83	0.05
8	Chandrasekharan and Rajagopalan (1987)	--	40×100	10	95.10	10	95.10	10	95.10	10	95.10	37.75	10	95.10	4.18
9	Chandrasekharan and Rajagopalan (1989, Fig. 1)	--	24×40	7	100.00	7	100.00	7	100.00	7	100.00	3.74	7	100.00	0.60
10	Chandrasekharan and Rajagopalan (1989, Fig. 2.2)	--	24×40	7	95.20	7	95.20	7	95.20	7	95.20	6.15	7	95.20	0.93
11	Chandrasekharan and Rajagopalan (1989, Fig. 2.3)	--	24×40	7	91.16	*	*	7	91.16	7	91.16	12.80	7	91.16	0.82
12	Chow and Hawaleshka (1992)	--	5×11	2	82.16	*	*	2	82.16	2	83.49	0.05	2	83.49	0.01
13	King and Nakornchai (1982)	--	5×7	2	91.18	2	91.18	2	91.18	2	91.18	0.01	2	91.18	0.01
14	Kumar and Vannelli (1987)	--	30×41	3	66.89	*	*	5	72.18	10	79.02	24.07	5	70.86	1.54
15	Kusiak and Chow (1987b)	--	7×8	3	82.50	*	*	3	82.50	4	88.78	0.05	3	82.50	0.01
16	Kusiak and Chow (1987b)	--	7×11	2	70.95	*	*	2	72.47	3	76.81	0.16	2	72.47	0.01
17	Kusiak and Cho (1992)	--	6×8	2	87.50	2	87.50	2	87.50	2	87.50	0.05	2	87.50	0.01
18	Kusiak (1992)	--	8×7	3	85.49	3	85.49	3	85.49	3	85.49	0.16	3	85.49	0.05

[a] "--" means the reference is the same as source in the left column

[b] Reference: Chen and Cheng (1995)

[c] "*" means that solution can not be obtained using basic ART1

[d] Computational time (in seconds)

Table 3.2. Computational results of test problems

No.	Problem Source	Non-neural algorithm reference[a]	Size ($m \times n$)	Reference solutions: Non-neural g	Reference solutions: Non-neural η (%)	Reference solutions: Basic ART1[b,c] g	Reference solutions: Basic ART1[b,c] η (%)	Reference solutions: Modified ART1[b] g	Reference solutions: Modified ART1[b] η (%)	OSHN: unrestricted number of cells g	OSHN: unrestricted number of cells η (%)	OSHN: unrestricted number of cells Time[d]	OSHN: restricted number of cells g	OSHN: restricted number of cells η (%)	OSHN: restricted number of cells Time[d]
19	Seifoddini (1989c, Fig.1)	--	9×12	3	89.41	3	89.41	3	89.41	3	89.41	0.16	3	89.41	0.01
20	Seifoddini (1989c, Fig.2)	--	9×12	3	92.48	3	92.48	3	92.48	3	92.48	0.16	3	92.48	0.01
21	Seifoddini (1989c, Fig.3)	--	9×12	3	93.75	3	93.75	3	93.75	3	93.75	0.16	3	93.75	0.01
22	Seifoddini (1989b)	Cheng (1992)	11×22	3	87.82	3	87.82	3	87.82	3	87.82	9.78	3	87.82	0.22
23	Seifoddini and Wolfe (1986)	--	8×12	3	85.53	3	85.53	3	85.53	4	90.93	0.11	3	85.53	0.05
24	Stanfel (1985)	Askin *et al.* (1991)	14×24	4	83.90	4	83.90	4	83.90	4	83.90	1.15	4	83.90	0.16
25	Waghodekar and Sahu (1984, Fig. 2)	--	5×7	2	85.62	2	85.62	2	85.62	2	85.62	0.05	2	85.62	0.01
26	Waghodekar and Sahu (1984, Fig. 3)	--	5×7	2	85.62	2	85.62	2	85.62	2	85.62	0.05	2	85.62	0.01
27	Waghodekar and Sahu (1984, Fig. 4)	--	5×7	2	77.10	*	*	2	77.10	2	79.61	0.05	2	79.61	0.01
28	Waghodekar and Sahu (1984, Fig. 5)	--	5×7	2	77.10	*	*	2	77.10	3	82.61	0.05	2	77.96	0.01

[a] "--" means that the reference is the same as source in the left column

[b] Reference: Chen and Cheng (1995)

[c] "*" means that solution can not be obtained using basic ART1

[d] Computational time (in seconds)

Table 3.2. (*continued*)

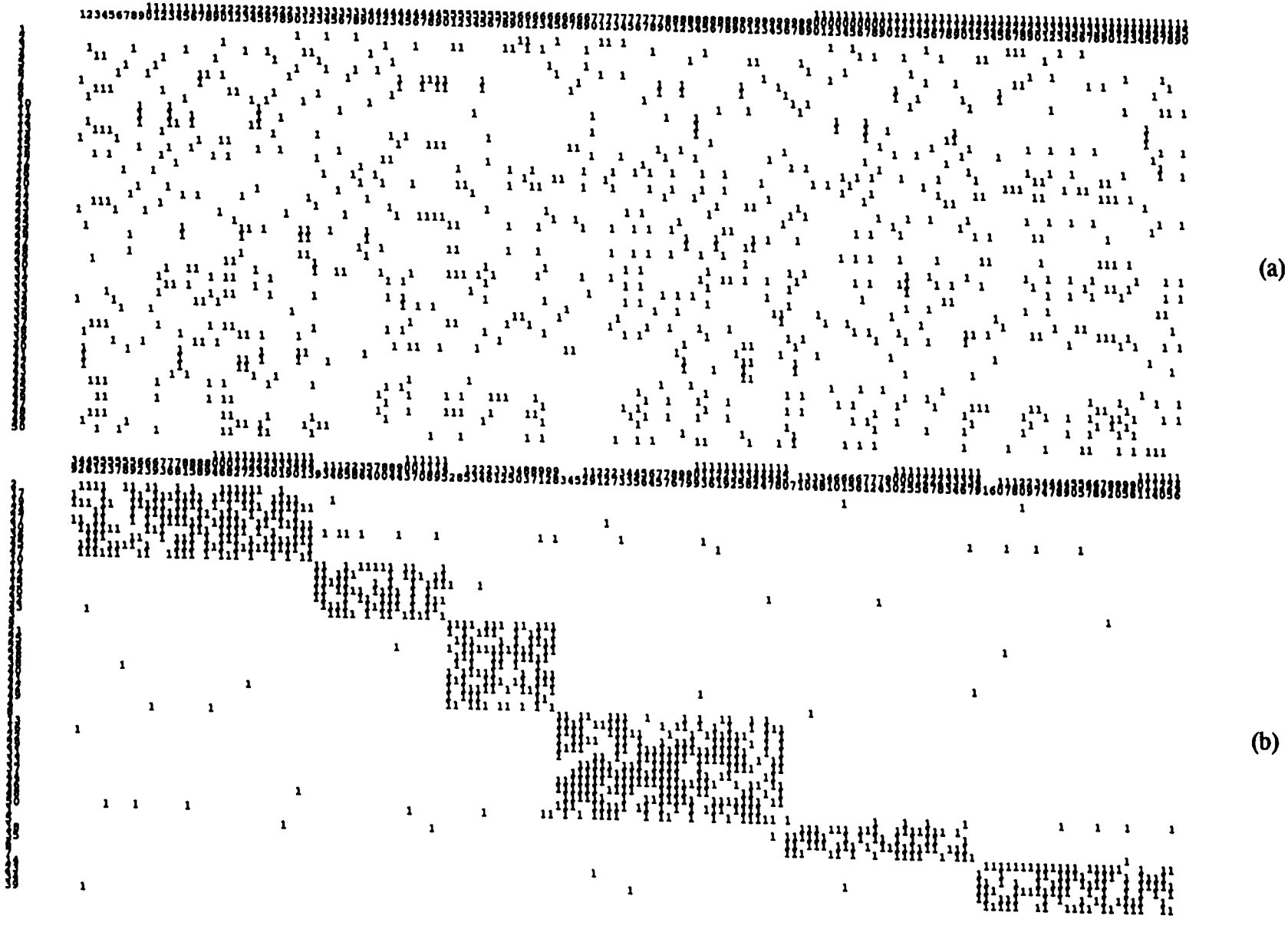


Figure 3.13. A 50×150 problem: (a) initial matrix, (b) final solution

3.5 Other Applications

The proposed algorithm can be applied to a variety of optimization problems. The following present the application of the proposed algorithm to some CMS design and planning problems.

3.5.1 Facility layout for cellular manufacturing

The layout problem in cellular manufacturing can be decomposed into two sub-problems: a) cell layout problem, and b) within-cell facility layout problem. The first sub-problem can be easily solved using traditional methods. However, within-cell facility layout problem requires further examination. If, as a result of grouping, R independent cells are obtained (i.e., there are no intercell moves), then the problem will be further decomposed into R machine layout problems that can be solved separately using traditional approaches. In reality, however, the machine cells are often not independent, i.e., some parts may have to be processed in more than one cell. This may lead to a large amount of intercell movements and hence the facility layout problem for different cells cannot be treated separately. The facility layout model is as follows:

Minimize

$$\sum_{x=1}^{M}\sum_{y=1}^{M}\sum_{s=1}^{S}\sum_{v=1}^{S} M_{xy}\, l_{sv}\, U_{xs}\, U_{yv} \tag{3.18}$$

subject to:

$$\sum_{x\in F_r} U_{xs} \le 1 \qquad \forall r,\ s\in S_r \tag{3.19}$$

$$\sum_{s\in S_r} U_{xs} = 1 \qquad \forall r,\ x\in F_r \tag{3.20}$$

$$U_{xs} = 0 \text{ or } 1 \qquad \forall r,\ x\in F_r,\ s\in S_r \tag{3.21}$$

The objective of the model is to minimize the total intercell and intracell travel distance. Constraint (3.19) specifies that each site should not be used by more than one machine, and Constraint (3.20) ensures that each machine is assigned to only one site. The last constraint is an integrality constraint. The above formulation is a 0-1 quadratic programming model which is computationally complex and cannot be solved in polynomial time for meaningful sized problems. It is therefore suggested that this problem be solved using the proposed neural network algorithm. The weight equation for the network connections is:

$$
\begin{aligned}
W_{xs,yt} = \; & -A\,\delta_{xy}(1-\delta_{st}) && \text{inhibitory connection within each row} \\
& -B\,\delta_{st}(1-\delta_{xy}) && \text{inhibitory connection within each column} \\
& -C && \text{global inhibition} \\
& -D\,M_{xy}\,d_{st} && \text{material handling term}
\end{aligned}
\qquad (3.22)
$$

With this weight equation, the same algorithm presented earlier in this chapter can be applied.

3.5.2 Grouping parts and tools for cellular manufacturing

Once machines and parts have been grouped, tool/part grouping becomes crucial. As explained earlier, a "1" element in the machine-part matrix indicates that a part needs to be processed on a machine. However, a single "1" element may represent several operations to be processed on the same machine, e.g., slotting, pocketing and drilling can all be done on a single milling machine. Each of these operations usually requires a different tool type. Due to limited capacity of the tool magazine, all the tools required for processing a part family may not be mounted simultaneously. Furthermore, the available copies of each tool type are also limited. The purpose of tool/part grouping is to reduce tool loading and searching time and to make better use of limited tool magazine

capacity as well as the available tools. These can be illustrated using the following example.

Example

Suppose the machine-part matrix shown in Figure 3.14 is obtained after the machine/part grouping. Now, a Machine in the first cell, say Machine 6, has to process the set of parts of the first family, i.e., {2,3,5,6,9}. The processing of each part may involve several operations using several different tools. Let $B_j=\{b_{xj}|x$: index of tools$\}$ denote the set of tools required by part type j. Assume that the processing of each part on machine 6 requires the following tool sets:

$$B_2=\{2,5,6,7\},\ B_3=\{1,2,6,7\},\ B_5=\{3,4,6\},\ B_6=\{1,2,5,7\},\ B_9=\{3,4,6\}$$

	Part								
Machine	2	3	5	6	9	1	4	7	8
3	1	1	1	1	1				
5	1	1	1		1				
6	1	1	1	1	1				
1						1	1	1	1
2						1	1	1	1
4						1	1	1	
7						1		1	1

Figure 3.14. A machine/part grouping example

It can be seen that the total number of tool types required to finish all the operations of the part family on machine 6 is 7. If the tool magazine on machine 6 can hold only 5 tools at a time, some tool changes are needed during the process of the part family. However, the number of tool changes can be minimized if tools and parts are grouped

properly. Such a grouping problem can be illustrated using a tool-part matrix for each machine. Figure 3.15 shows a tool-part matrix and the grouping result for machine 6 of the first cell.

The grouping result indicates that the part family can be further decomposed into two sub-families, and the tool set can be divided into two groups. Thus, each sub-family is processed by one tool group that can be completely mounted on the tool magazine. It should be noted that tool 6 is duplicated in both groups in order to make the groups independent.

	Part				
Tool	2	3	5	6	9
1		1		1	
2	1	1		1	
3			1		1
4			1		1
5	1			1	
6	1	1	1		1
7	1	1		1	

(a) Tool-part matrix

	Part				
Tool	2	3	6	5	9
1		1	1		
2	1	1	1		
5	1		1		
6	1	1			
7	1	1	1		
3				1	1
4				1	1
6				1	1

(b) Grouping result

Figure 3.15. The tool-part matrix and grouping result for tools and parts on machine 6

The part/tool grouping model can be expressed as follows:

Maximize
$$\sum_{r=1}^{M} \eta(r) \tag{3.23}$$

subject to:

$$\sum_{x} U_{xr} \le T \qquad \forall r \tag{3.24}$$

$$\sum_{r} U_{xr} \le n_x \qquad \forall x \tag{3.25}$$

where $\eta(r)$ has the same structure as that of the grouping efficiency defined by Equation 3.1 when the number of groups is r.

The objective of the above model is to maximize the grouping efficiency. The first constraint represents the capacity limitation of the tool magazine, and the second imposes the restriction on the number of available duplicates of each tool type. Such a tool/part grouping problem can be easily solved using the proposed algorithm.

CHAPTER 4

COMPREHENSIVE MACHINE CELL/PART FAMILY FORMATION

4.1 Background

In Chapter 3, a neural network approach was proposed to efficiently solve the binary machine/part grouping problem and a variety of clustering problems. As stated earlier, in binary grouping, the differences in processing times and lot sizes are ignored since the part demands are not available at the early design stage. However, if an accurate forecast of demand can be made, then workload balance and cell utilization can be improved by taking the processing times and lot sizes into consideration. In this chapter, the processing times, lot sizes and machine capacities will also be considered in the CMS design using two different approaches. The first approach is based on a search method known as *simulated annealing*, and the second one is an objective-guided neural network approach based on the OSHN introduced in Chapter 3.

In order to evaluate the solution quality during the search, a grouping measure is required. The grouping efficiency η, discussed in Section 3.2, cannot be used since it treats all the operations equally even though they are associated with different processing times and lot sizes. The result of maximizing η is the maximized concentration of 1's in the diagonal clusters and minimized number of exceptional elements. In other words, maximizing η is in effect to drive as many 1's into diagonal clusters as possible. However, the same η may be achieved by different machine cell/part family compositions, or different distribution patterns of 1's. As the procedure of maximizing

η does not specify how the 1's are distributed in the clusters, maximizing η alone may not provide a desirable solution. Furthermore, as processing times and machine capacities are not considered, a feasible solution cannot be guaranteed. To illustrate, consider the machine-part matrix in Figure 4.1(a).

Machine \ Part	1	2	3	4
1	1	1	1	
2	1		1	1
3		1	1	1
4	1	1		1

Machine \ Part	1	2	3	4
1	1	1	2	
2	2		3	1
3		2	1	3
4	1	2		2

(a) Machine-part incidence matrix and processing times

Machine \ Part	1	4	2	3
2	1	1		1
4	1	1	1	
1	1		1	1
3		1	1	1

Machine \ Part	1	4	2	3
2	2	1		3
4	1	2	2	
1	1		1	2
3		3	2	1

(b) A grouping solution and the associated processing times with $\eta=0.75$ and $\lambda=12/9$

Machine \ Part	2	4	1	3
3	1	1		1
4	1	1	1	
1	1		1	1
2		1	1	1

Machine \ Part	2	4	1	3
3	2	3		1
4	2	2	1	
1	1		1	2
2		1	2	3

(c) An alternative solution with $\eta=0.75$ and $\lambda=17/4$

Figure 4.1. Different solutions for a grouping problem when processing times are considered

As shown in this figure, the processing time associated with each "1" varies considerably. Figure 4.1(b) shows a solution to this problem with a grouping efficiency η=0.75, whereas an alternative solution can be obtained with the same grouping efficiency but different within-cell workloads as shown in Figure 4.1(c). The within-cell workload can be measured by a *workload ratio*, λ, defined as follows:

$$\lambda = \frac{\text{Total in-cell processing time}}{\text{Total out-cell processing time}}$$

This ratio measures the level of in-cell machine utilization. The higher the workload ratio, the higher the in-cell workload utilization will be. It can be observed that for the first solution (Figure 4.1(b)) the workload ratio is 12/9, but for the second solution (Figure 4.1(c)) this ratio increases to 17/4. Thus, when different processing times are involved, η cannot reflect the real cell utilization and using η alone is not adequate.

In view of the above, a *generalized grouping efficiency* is proposed to accommodate the effect of processing times and lot sizes. The details are explained in the next section.

4.2 Generalized Grouping Efficiency

The generalized grouping efficiency, η_g, is a revised form of η that has accommodated the effects of the processing times and lot sizes. η_g is given as follows:

$$\eta_g = q_g \eta_d + (1-q_g)\eta_o \tag{4.1}$$

where η_d measures the density of processing times inside cells and is given below:

$$\eta_d = \frac{t_d}{\sum_{r=1}^{R} M_r \sum_{\substack{j=1 \\ j\in\Omega_r}}^{N} L_j \, t_j^{max}} \tag{4.2}$$

and η_o reflects the sparsity of processing times outside cells which is given by:

$$\eta_o = 1 - \frac{t_o}{m\sum_{j=1}^{N} L_j t_j^{max} - \sum_{r=1}^{R} M_r \sum_{\substack{j=1 \\ j\in\Omega_r}}^{N} L_j t_j^{max}} \tag{4.3}$$

q_g is the weighting factor. Following the suggestion of Kumar and Chandrasekharan (1990), a counterpart of Equation 3.4 is defined as follows:

$$q_g = \frac{\sum_{r=1}^{R} M_r \sum_{\substack{j=1 \\ j\in\Omega_r}}^{N} L_j t_j^{max}}{m\sum_{j=1}^{N} L_j t_j^{max}} \tag{4.4}$$

Unlike η, the generalized grouping efficiency, η_g, does not treat all the operations equally. The priority of an operation depends on its processing time. For the solutions shown in Figures 4.1(b) and 4.1(c), the generalized grouping efficiencies, η_g values, are respectively 0.625 and 0.825, and thus solution 4.1(c) may be preferable to solution 4.1(b). It is worthwhile to note that if all the processing times and lot sizes are equal, then η_g reduces to η.

Similar to η, the calculation of η_g requires to know the configurations of both machine groups and part families. The machine groups will be identified using one of the two approaches presented later in this chapter. Each of these two approaches deals with machine grouping differently. The part assignment method, however, is the same for both and is described below.

4.3 Part Assignment

The task of part assignment is to match parts with the obtained machine cells. The

matching of parts and machine cells is done using the following membership index:

$$D_{rj} = \frac{m_{rj}}{M_r} \cdot \frac{m_{rj}}{m_j} \cdot \frac{h_{rj}}{H_j} \tag{4.5}$$

The above membership index measures the degree of belongingness of each part type j to an existing machine cell r. This membership index is composed of three components and each of them reflects one aspect of belongingness. The first component indicates the proportion of machines of cell r required by part type j. The second one is the ratio of number of machines of cell r required by part type j to the total number of machines required by part type j, and the third shows the proportion of processing time of part type j that can be performed in cell r. D_{rj} ranges between 0 and 1. $D_{rj}=1$ means a perfect match between part type j and cell r (i.e., all operations of part j can be completely processed by cell r), and $D_{rj}=0$ indicates a complete mismatch between part type j and cell r (i.e., none of part j's operations can be processed by cell r). Therefore, part type j should be assigned to a cell r^* that yields the highest membership level, i.e.,

$$D_{r^*j} = \max_r \{D_{rj} \mid r=1,\ldots,R\} \tag{4.6}$$

For example, if in the iterative process of the problem shown in Figure 4.1(a) the following machine cell configuration is obtained after machine grouping:

MC-1: {3,4}, MC-2: {1,2}

we will have $M_1=M_2=2$ since each cell contains 2 machines. To identify associated part families, we need to calculate D_{rj} for all parts and current cells. The results are summarized in Table 4.1.

Part j	m_j	H_j	Cell 1			Cell 2		
			m_{1j}	h_{1j}	D_{1j}	m_{2j}	h_{2j}	D_{2j}
1	3	4	1	1	0.04	2	3	0.50*
2	3	5	2	4	0.53*	1	1	0.03
3	3	6	1	1	0.03	2	5	0.56*
4	3	6	2	5	0.56*	1	1	0.03

Table 4.1. Calculation summary for part assignment

The asterisk indicates the largest D_{ij} for a part. Therefore, the following part family configuration is obtained:

PF-1: {2,4}, PF-2: {1,3}

The above part assignment will be used in both machine grouping approaches. As stated earlier, the first approach is based on simulated annealing search method and the second uses the ortho-synapse Hopfield network (OSHN) approach in conjunction with an objective-guided algorithm. Both approaches aim at the maximization of the generalized grouping efficiency, η_g.

In many machine grouping methods, it is often assumed that only one machine of each type is available. Dropping this assumption may improve the solution quality if bottleneck machines exist. In reality, each machine has a limited capacity, and the processing time requirement beyond its capacity should be performed by duplicated machines. Furthermore, considering the possibility of duplication will help reduce the volume of intercell movement and improve the workload balance. Such a decision is also included in both approaches. The details of these two approaches are discussed in the following sections.

4.4 Simulated Annealing

Simulated annealing (SA), initially introduced by Kirkpatrick *et al.* (1983), is a random search technique capable of escaping local optima by considering a transition probability. SA needs to be initialized by a seed solution known as the *current state*, and at every iteration it is accompanied by two other states, *neighbouring* and *optimal* states. Initially, all the three states are identical. Once the search begins, a neighbouring state is generated by making a small change in current state. The quality of each state is measured by a cost function. If the neighbouring cost is less than the optimal cost, both the current and optimal states will take the neighbouring state for the next iteration. If the neighbouring cost is less than current cost but not less than optimal cost, only current state is updated. In case that neighbouring cost is greater than current cost, there is still a chance for current state to transit to neighbouring state depending on the transition probability. If the transition probability is greater than a uniformly random generated number, then the transition to the worse state will take place; otherwise, the iteration number will be incremented and another neighbour will be generated. Unlike some other search methods, in simulated annealing only one neighbouring solution at each iteration is generated and compared to the current solution. This results in less computational time for each iteration, but possibly more iterations. The magnitude of the transition probability depends upon the difference between neighbouring and current costs, and also upon a positive control parameter called *temperature*. A higher temperature, θ, or a smaller difference between two costs, ΔC, yields a higher transition probability. If the temperature gradually decreases as the number of iterations increases, a cooling process known as *annealing* is incurred. This stochastic process continues until a maximum allowed number of iterations or a maximum allowed number of identical optimal states

is reached.

Denoting β_t, γ_t, and π_t as the current, neighbouring, and optimal states at iteration t, respectively, and $C(X)$ as the cost of state X, the probability of transition from current state to neighbouring state at iteration t is then defined as:

$$Pr(\beta_{t+1} = \gamma_t) = min\left\{1, \exp\left(-\frac{C(\gamma_t) - C(\beta_t)}{\theta_t}\right)\right\} \tag{4.7}$$

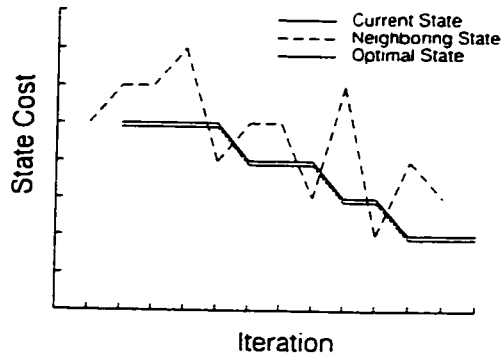
where θ_t represents the temperature at iteration t. When the above probability is less than 1, it indicates the possibility of transition from a lower cost to a higher cost. Therefore, this probability is also called *uphill transition probability*. By decreasing the temperature while the iteration number, t, is increasing, the uphill transition probability declines toward zero. When t approaches a sufficiently large number, this probability becomes zero, i.e., $\lim_{t\to\infty} Pr(\beta_{t+1}=\gamma_t)=0$, and at this point, the stochastic search process becomes a deterministic iterative search process. The temperature declining is performed using a function known as *annealing schedule* or *cooling schedule*. In this thesis the following annealing schedule is used to decrement the temperature:

$$\theta_t = \frac{\theta_0}{1+\ln t} \tag{4.8}$$

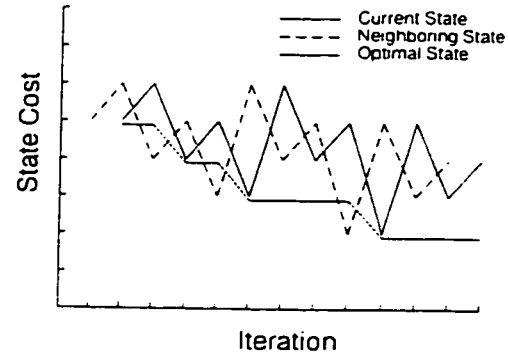
The initial temperature, θ_0, is a parameter which controls the highest point of the transition probability. The closer to zero the θ_0 value, the smoother the stochastic search process will be. Again at the limit point, $\theta_0=0$, the process becomes deterministic.

Our computational experience shows that θ_0 is an important parameter and should be selected carefully. On one hand, if a very low θ_0 is selected, the annealing schedule will quickly reach the freezing point, and consequently the stochastic search process

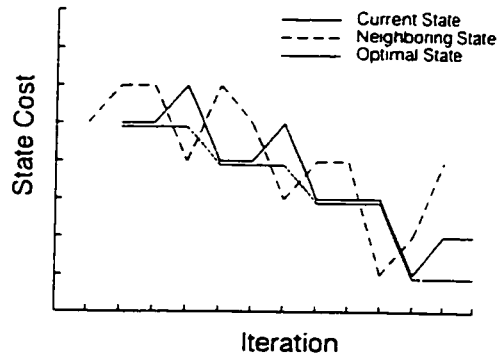
becomes deterministic. Once it becomes deterministic, the current state will never transit to a higher cost (Figure 4.2(a)), and thus there will be no chance to escape a local minimum. On the other hand, if θ_0 is too high, the chance of transition from current solution to a worse solution will increase considerably. Thus the current state will simply follow the neighbouring state pattern (Figure 4.2(b)), and hence the search process deteriorates an unguided random process though this process may not last too long as per Equation 4.8. Therefore, a good initial temperature should be selected in such a way that the current state will have a better chance of transition to a higher cost (Figure 4.2(c)).



(a) Effect of an unreasonably low θ_0



(b) Effect of an unreasonably high θ_0



(c) A good initial temperature

Figure 4.2. The impact of the initial temperature on the state cost

Following the idea of the simulated annealing, a proper initial temperature is the one that yields a probability of close to 1 for the first iteration. Thus, by inspecting Equation 4.7, the following range is suggested as a guideline for selecting θ_0.

$$1.45\ \Delta C_{max} \leq \theta_0 \leq 100\ \Delta C_{max}$$

where ΔC_{max} is the maximum expected ΔC during a search process. This guideline ensures that for the first iteration if the worst case (i.e., $\Delta C = \Delta C_{max}$) occurs, the probability of transition will stay in the range between 0.5 and 0.99. This range will be adjusted as the iteration number increases.

4.4.1 Application of SA to comprehensive machine/part grouping

Liu and Wu (1993), and Chen *et al.* (1995) have applied simulated annealing to the machine grouping problem. However, their applications are based on the binary machine-part matrix, and thus the processing time and machine capacity are not considered. In this study, both processing time and machine capacity are incorporated. The generalized grouping efficiency, η_g, is used as the objective function. To apply the SA approach, a cost function for state evaluation is defined as follows:

$$C(X) = 1 - \eta_g(X) \tag{4.9}$$

where $\eta_g(X)$ is the performance measure of state X given by Equation 4.1. Since η_g ranges between 0 and 1, the state cost, $C(X)$, is also in the range of 0 and 1. To initialize the states, a feasible seed solution is needed. The feasible seed solution may be constructed using any existing binary matrix based method. In this study, we generate the initial machine groups using the OSHN approach discussed in Chapter 3. The reason for using this method is that it can quickly find a good solution with a

higher η value. The OSHN solution is then improved using the SA algorithm to maximize η_g. Figure 4.3 illustrates this solution procedure.

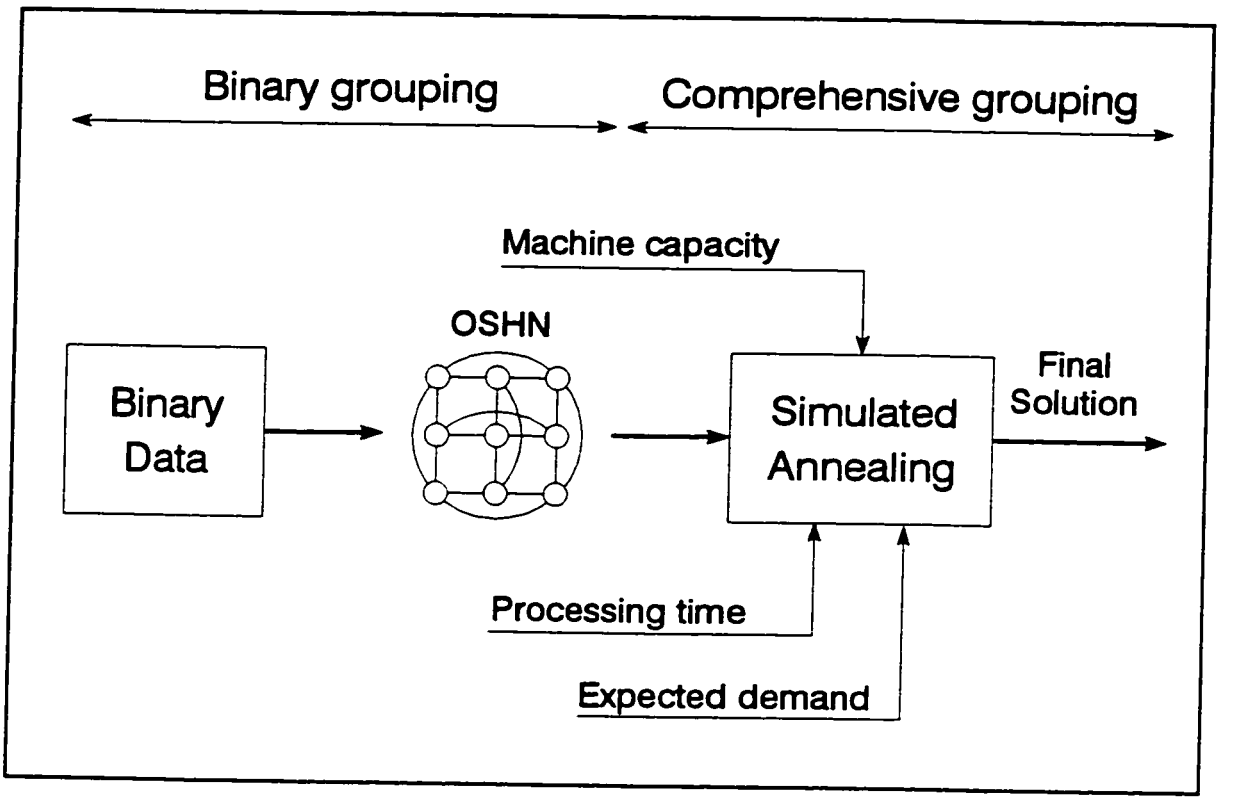


Figure 4.3. The solution procedure for application of SA to comprehensive machine grouping problem

At a given iteration of the SA, the neighbouring state is generated based on the current state. In our application, a neighbouring solution is constructed from the current state by reassigning a machine from its current cell to another cell. Every neighbouring solution has to be feasible in terms of machine capacity. Therefore, machine capacities are checked when a neighbouring solution is generated. If a capacity constraint is violated, machine duplication will be considered.

The simulated annealing algorithm for machine cell/part family formation problem is presented below:

Algorithm 4.1

Step 1 Read θ_0, P, and t_{max}, and set $t \leftarrow 0$.

Step 2 Initialize current state, β_0, and set $\pi_0 \leftarrow \beta_0$ and $\gamma_0 \leftarrow \beta_0$.

Step 3 Set $t \leftarrow t+1$.

Step 4 Generate a feasible neighbouring state, γ_t, based on β_t.

Step 5 Calculate cost for all states, i.e., $C(\beta_t)$, $C(\gamma_t)$, and $C(\pi_t)$.

Step 6 If $C(\gamma_t) < C(\pi_t)$, then set $\pi_t \leftarrow \gamma_t$ and go to Step 10; else, continue.

Step 7 Update the temperature according to Equation 4.8 and calculate the transition probability, $Pr(\beta_{t+1}=\gamma_t)$, using Equation 4.7.

Step 8 Generate a uniform random number, α, from the interval [0,1].

Step 9 If $Pr(\beta_{t+1}=\gamma_t) < \alpha$, go to Step 11; else, continue.

Step 10 Set $\beta_{t+1} \leftarrow \gamma_t$.

Step 11 If $t=t_{max}$, stop; else, go to Step 3.

Illustrative example

The above algorithm has been coded in C, and a notable problem initially studied by Burbidge (1975) is selected to illustrate the application of the proposed algorithm. The binary version of this problem (Figure 3.7) consists of 16 machine types and 43 part types. Rao and Gu (1995) further examined this problem by assigning a value t_{ij} (processing time) to each nonzero entry of this problem to test their neural network approach. Here, their machine-part matrix is adopted and reproduced in Figure 4.4. The maximum number of SA iterations is set to 2000. The seed solution is generated using

the ortho-synapse Hopfield network method which takes only 0.22 second on a 486-33 MHz computer. It took about 55 seconds for SA to complete 2000 iterations, and the final solution obtained at iteration 1942. The final solution comprises of 5 cells and 19 machines, and yields a generalized grouping efficiency, η_g=0.8771.

Rao and Gu (1995) considered different objectives in their study. However, to evaluate the quality of the above solution, the solution reported by Rao and Gu (1995) is listed as follows:

Number of cells: 4

Machine groups: {1,2,3,6,8,9,14,16},{8,11,12,13},{6,7,8,10},{4,5,6,8,11,15}

Part families: {2,4,6,7,10,17,18,28,32,34,35,36,37,38,40,42},{3,11,20,22,24, 27,30},{1,12,13,25,26,31,39},{5,8,9,14,15,16,19,21,23,29, 33,41,43}

Grouping efficiency, η=0.8562

Generalized grouping efficiency, η_g=0.8103

The above reference solution has 4 cells, while our final solution has 5. As both the Hopfield network method and the SA algorithm are able to control the maximum number of cells, the maximum number of cells is set at 4 and the computation is repeated for a fair comparison with Rao and Gu's result. In this case, the final solution is obtained at iteration 890 which has the following composition:

Number of cells: 4

Machine groups: {4,5,6,8,15},{3,8,11,12,13,14},{6,7,8,10},{1,2,6,9,16}

Part families: {5,8,9,14,15,16,19,21,23,29,33,4143},{3,11,20,22,24,27,

	Machine															
Part	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	4.83	0.33	2.39	0	4.55	0	0	0	0	0	0
2	0	1.76	0	0	0	4.66	0	3.27	0.11	0	0	0	0	2.56	0	1.01
3	0	0	0	0	0	0	0	4.70	0	0	1.02	0	1.89	0	0	0
4	0	0	0	0	0	0	0	0	3.97	0	0	0	0	0	0	0
5	0	0	0	1.44	1.34	0	0	0	0	0	0	0	0	0	1.78	0
6	0	0	0	0	0	4.64	0	0	0	0	0	0	0	3.52	0	0
7	0	0	3.69	0	0	4.01	0	0	0	0	0	0	0	0	0	4.58
8	0	0	0	0	2.56	3.81	0	2.28	0	0	0	0	0	0	0	0
9	0	0	0	2.36	1.50	0	0	3.06	0	0	0.37	0	0	0	0	0
10	0	2.49	0	0	0	0	0	0	1.10	0	0	0	0	0	0	0.21
11	0	0	0	0	0	0	0	2.82	0	0	0	3.49	0	0	0	0
12	0	0	0	0	0	4.75	0	4.58	0	3.15	0	0	0	0	0	0
13	0	0	0	0	0	3.03	4.69	0	0	0.71	0	0	0	0	0	0
14	0	0	0	4.04	4.39	1.73	0	0	0	0	0	0	0	0	0.93	0
15	0	0	0	0	3.35	0	0	3.17	0	0	0	0	0	0	0	0
16	0	0	0	0	2.27	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0.14	0	0	3.82	0	0	0	0	0	0	0	0.78	0	0
18	0	0	0	0	0	0	0	0	3.83	0	0	0	0	0	0	2.83
19	0	0	0	0.36	1.38	1.64	0	2.64	0	0	0	0	0	0	3.74	0
20	0	0	0	0	0	0	0	3.14	0	0	0.70	0	0	0	0	0
21	0	0	0	4.10	0.63	0	0	1.80	0	0	0	0	0	0	4.31	0
22	0	0	0	0	0	0	0	0	0	0	0	3.45	0	0	0	0
23	0	0	0	0.29	4.07	3.04	0	3.44	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	2.09	0	0	2.72	4.16	1.13	0	0	0
25	0	0	0	0	0	0	2.11	0	0	0.89	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	2.06	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0.47	0	0	4.06	4.33	0	0	0	0
28	0	0.60	0	0	0	0	0	2.88	0.11	0	0	0	0	0	0	0
29	0	0	0	4.43	0.71	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0.47	0.81	0	0	0	0
31	0	0	0	0	0	0	0	2.35	0	3.11	0	0	0	0	0	0
32	0	4.55	0	0	0	0.49	0	0	3.82	0	0	0	0	0	0	3.65
33	0	0	0	0	1.11	0.62	0	0	0	0	0	0	0	0	2.96	0
34	0	0	4.57	0	0	3.91	0	0	0	0	0	0	0	0	0	0
35	0	0	2.03	0	0	0	0	0	0	0	0	0	0	2.60	0	0
36	0	0	4.36	0	0	0	0	0	0	0	0	0	0	0	0	0
37	4.12	0.33	0	0	0	3.51	0	0.25	2.44	0	0	0	0	0	0	4.40
38	0	2.32	0	0	0	0	0	2.90	3.47	0	0	0	0	0	0	1.65
39	0	0	0	0	0	3.51	0	0	0	1.35	0	0	0	0	0	0
40	0	1.76	0	0	0	5.94	0	0	2.06	0	0	0	0	0	0	0
41	0	0	0	0	2.23	0	0	3.75	0	0	0	0	0	0	4.41	0
42	0.34	3.30	0	0	0	4.89	0	0	4.16	0	0	0	0	0	0	1.95
43	0	0	0	0	1.01	4.78	0	4.91	0	0	0	0	0	0	0.58	0

Figure 4.4. Machine-part matrix consisting of 16 machine types and 43 part types (entries are t_{ij}'s)

30,35,36},{1,6,12,13,17,25,26,28,31,34,39},{2,4,7,10,
18,32,37,38,40,42}

Grouping efficiency, η=0.8709

Generalized grouping efficiency, η_g=0.8406

It is seen the solution from the SA algorithm is preferred in terms of both η and η_g. Furthermore, in the reference solution 22 machines are needed, whereas SA solution requires only 20 machines.

4.4.2 Discussion

(1) Less duplicated machines

The comparison of the proposed SA algorithm and the reference method indicates that the SA algorithm can find more alternative solutions and better solutions with less duplicated machines. As compared to 22 machines used in the reference solution, the SA approach found a solution that uses only 19 machines when the number of cells is pre-specified, and a solution with 20 machines if this number is not pre-determined.

(2) Higher in-cell machine utilization

By using the generalized grouping efficiency, the proposed algorithm can automatically assign higher priorities to those operations with longer processing times and larger lot sizes. As a result, the high-load operations will be concentrated in cells and the low-load operations will be driven out of cells. This load distribution not only provides higher in-cell machine utilization rate but also facilitates later operation scheduling.

(3) Improved workload balance

The computational results also reveal that, in the solution obtained using the SA approach, the workload is better balanced compared to the reference solution. The standard deviation of workload in the solution obtained using the proposed SA algorithm is 7.2, while this value for the reference solution is 8.6.

(4) Better seed solution

The final solution of the SA approach, like any other search methods, depends on the seed solution. In the problem under investigation, the seed solution is a feasible solution that can be generated either randomly or using any binary grouping method. Though a random seed solution can be quickly generated, its quality is usually very poor in comparison to the one obtained by a binary grouping method. Therefore, the seed solution for the proposed SA approach will be obtained using the OSHN algorithm.

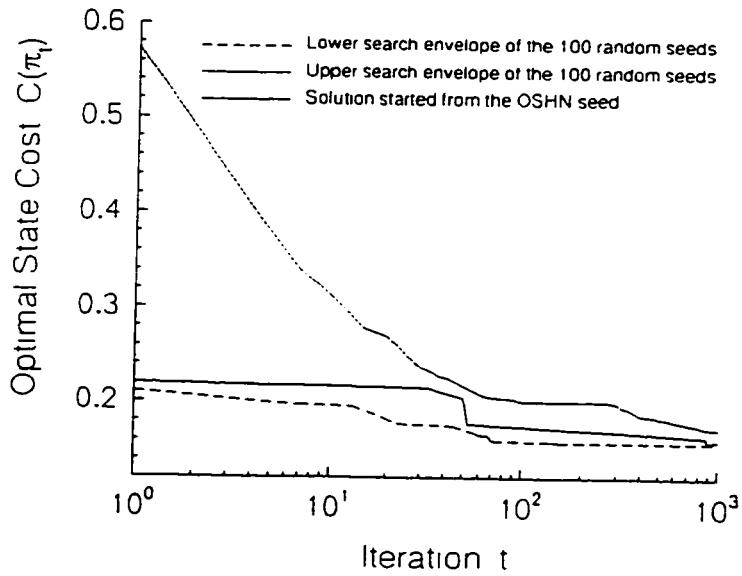


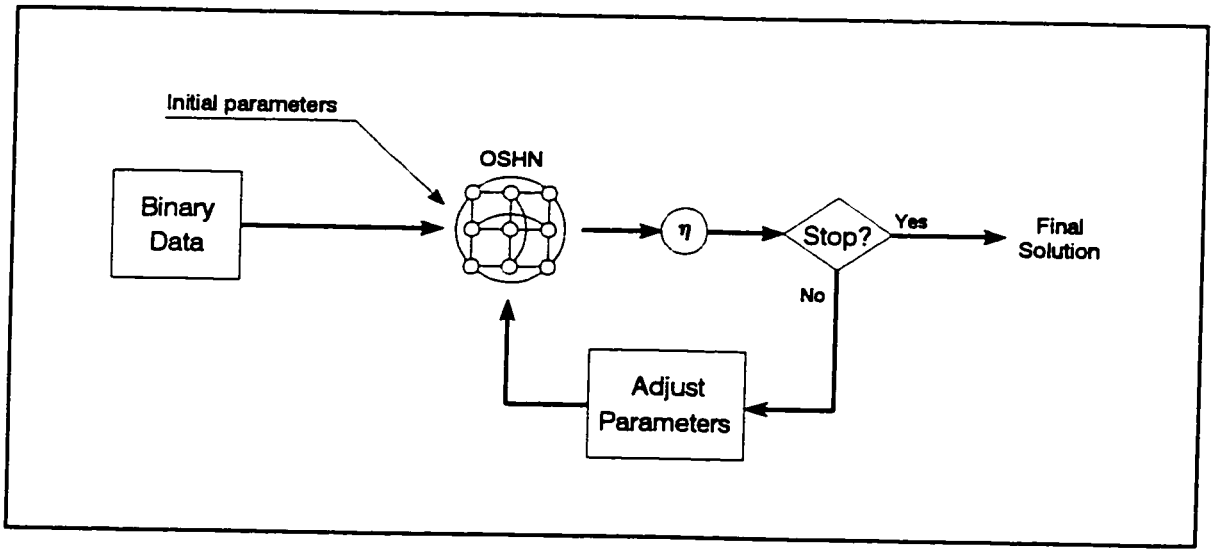
Figure 4.5. SA performance starting from OSHN output compared to the upper and lower envelopes of 100 random seed solutions

To compare the search performance between the OSHN seed solution and random generated seeds, 100 seed solutions were generated randomly for the above example problem, and then the SA algorithm was run for 1000 iterations for the OSHN seed and each random seed solution. The results are summarized in Figure 4.5. In this figure, the lower and upper lines respectively represent the lower and upper search envelopes based on the 100 random seed solutions. It is shown that the convergence curve based on the OSHN seed is very close to the lower envelope, i.e., the best search path obtained based on all the 100 random seeds.

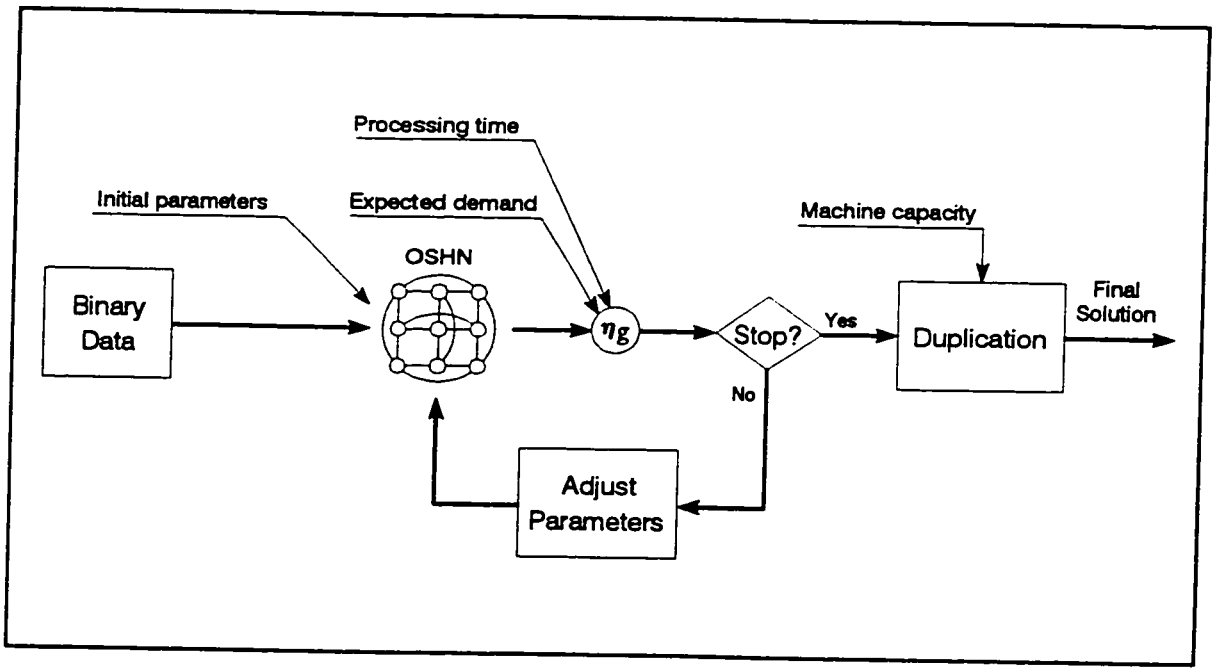
4.5 The Ortho-Synapse Hopfield Network for Comprehensive Grouping

In Chapter 3, the OSHN, a new structure of the Hopfield network, has been proposed to solve binary grouping problems. As discussed earlier, the final solution of the OSHN depends on three parameters of the network. In order to find a set of parameters that yields the maximum grouping efficiency η, the OSHN has been embedded in an objective-guided algorithm (Algorithm 3.4) . As shown in Figure 4.6(a), this algorithm tunes the parameters according to the intermediate results.

To solve the comprehensive grouping problems, the same methodology can be used, but instead of grouping efficiency η, the generalized grouping efficiency η_g is used as the objective of the algorithm. After converging to the final solution, a separate module is used to check the capacity constraints and perform duplication of high load machines as necessary. This methodology is illustrated in Figure 4.6(b). To distinguish the application of the OSHN approach to the binary grouping from the comprehensive grouping, the later is denoted as $OSHN_g$.



(a) Binary grouping (OSHN)



(b) Comprehensive grouping ($OSHN_g$)

Figure 4.6. The major steps of the objective-guided algorithm

4.5.1 Application of OSHN$_g$ to comprehensive machine/part grouping

As stated above, the OSHN$_g$ approach is similar to the OSHN, but instead of η, η_g is used to guide the algorithm. The associated algorithm is summarized as follows.

Algorithm 4.2

Step 1 Set $A \leftarrow 0$, $C \leftarrow 1$, $D \leftarrow 1$, $\eta_g \leftarrow 0$, $CTR \leftarrow 0$, read ΔA, ΔC, ΔD, u, NS, and compute S_{xy}.

Step 2 If the maximum number of cells is pre-specified as K, set $V \leftarrow K$; otherwise, set $V \leftarrow M$.

Step 3 Calculate $\omega_{xr,yq}$ using Equation 3.13.

Step 4 Update the network using Equations 3.6 and 3.17.

Step 5 If the state of network is the same as that of the last iteration, go to the next step; otherwise, go to Step 4.

Step 6 If there is any unassigned machine, say machine y, find $\hat{S}_{yr}=max\{\hat{S}_{yq} | q=1,\ldots,R\}$ and assign machine y to cell r; otherwise, go to the next step.

Step 7 Identify part families based on the membership measure defined in Equations 4.5 and 4.6.

Step 8 If duplicated machine assignments are observed and they are not allowed, set $A \leftarrow A+\Delta A$ and go to Step 4; otherwise, go to the next step.

Step 9 If the number of small cells is greater than u, set $D \leftarrow D+\Delta D$ and go to Step 4; otherwise, go to the next step.

Step 10 If all machines have been grouped into a single cell, set $C \leftarrow C+\Delta C$ and go to Step 4; otherwise, go to the next step.

Step 11 Calculate the generalized grouping efficiency, φ_g, obtained in the current

iteration. If $\varphi_g > \eta_g$, set $\eta_g \leftarrow \varphi_g$, $D \leftarrow D + \Delta D$, $CTR \leftarrow 0$ and go to Step 4; otherwise, $CTR \leftarrow CTR + 1$, and go to the next step.

Step 12 If $CTR < NS$, go to Step 4; otherwise, discard the current solution and accept the solution corresponding to η_g as the final solution.

Step 13 Check the capacity constraints for all machines. If the load on any machine exceeds the capacity limit, then duplicate that machine to fulfil the workload.

Illustrative example

The above algorithm has been coded in C and tested on the example problem shown in Figure 4.4. It took only 0.60 seconds for the $OSHN_g$ to stop at the final solution with the generalized grouping efficiency $\eta_g = 0.8604$. Similar to the solution obtained using SA, the final solution of $OSHN_g$ uses 19 machines in 5 cells. Comparison of SA and $OSHN_g$ on this example problem shows that the SA approach is able to find a better solution (i.e., 0.8771 vs. 0.8604), but the $OSHN_g$ is much faster. In order to get a better insight into the performance of these two approaches more problems are tested in the following section.

4.6 Discussion

In order to compare the performance of the SA and the $OSHN_g$, the 28 binary problems reported in Table 3.2 have been used for testing with processing times and lot sizes being randomly assigned to all operations and parts of each problem. The maximum number of iterations for SA algorithm is set to 2000. The computational results are summarized in Table 4.2.

No.	Problem Source	Size ($m \times n$)	Number of cells	SA		$OSHN_g$	
				η_g (%)	Time[a]	η_g (%)	Time[a]
1	Askin *et al.* (1991)	10×15	3	83.23	12.86	82.43	0.05
2	Boe and Cheng (1991)	20×35	4	83.37	44.67	83.03	0.71
3	Burbidge (1975)	16×43	5	88.27	51.43	84.86	0.60
4	Burbidge (1975) (excluding machines 6 and 8)	14×43	5	89.75	42.86	89.19	0.27
5	Burbidge (1975) (duplicating machines 6 and 8)	22×43	5	88.10	60.71	87.09	0.60
6	Carrie (1973)	20×35	4	86.79	42.86	86.79	0.60
7	Chandrasekharan and Rajagopalan (1986)	8×20	3	85.73	14.78	85.73	0.11
8	Chandrasekharan and Rajagopalan (1987)	40×100	10	90.97	270.82	94.81	4.67
9	Chandrasekharan and Rajagopalan (1989, Fig. 1)	24×40	7	92.90	72.75	94.71	0.55
10	Chandrasekharan and Rajagopalan (1989, Fig. 2.2)	24×40	7	91.77	69.18	93.56	1.04
11	Chandrasekharan and Rajagopalan (1989, Fig. 2.3)	24×40	7	90.72	69.51	92.15	1.26
12	Chow and Hawaleshka (1992)	5×11	2	74.20	5.16	74.20	0.01
13	King and Nakornchai (1982)	5×7	2	81.96	3.35	81.96	0.01
14	Kumar and Vannelli (1987)	30×41	3	85.62	72.25	84.12	1.15
15	Kusiak and Chow (1987b)	7×8	3	85.33	5.00	84.47	0.05
16	Kusiak and Chow (1987b)	7×11	2	84.62	7.47	77.93	0.05
17	Kusiak and Cho (1992)	6×8	2	78.37	4.12	78.37	0.01
18	Kusiak (1992)	8×7	3	84.91	5.38	84.91	0.05
19	Seifoddini (1989c, Fig.1)	9×12	3	81.49	10.11	81.25	0.05
20	Seifoddini (1989c, Fig.2)	9×12	3	86.20	9.94	86.20	0.05
21	Seifoddini (1989c, Fig.3)	9×12	3	86.50	9.89	86.50	0.05
22	Seifoddini (1989b)	11×22	3	81.34	20.11	81.08	0.11
23	Seifoddini and Wolfe (1986)	8×12	3	80.50	9.40	71.50	0.05
24	Stanfel (1985)	14×24	4	85.65	22.47	85.58	0.16
25	Waghodekar and Sahu (1984, Fig. 2)	5×7	2	75.48	3.46	75.48	0.01
26	Waghodekar and Sahu (1984, Fig. 3)	5×7	2	74.14	3.46	74.14	0.01
27	Waghodekar and Sahu (1984, Fig. 4)	5×7	2	69.41	3.90	60.09	0.01
28	Waghodekar and Sahu (1984, Fig. 5)	5×7	2	81.44	3.79	62.62	0.01

[a] Computational time (in seconds) on a 486-33 MHz PC

Table 4.2. The computational results of SA and $OSHN_g$

The results indicate that, the SA approach outperforms the $OSHN_g$ in terms of the solution quality. Out of the 28 test problems, both approaches yield equal η_g in 10 instances, and the SA obtains a higher η_g value in 14 cases, whereas the $OSHN_g$ gives

a better solution for only 4 problems. In terms of the computational time, however, the $OSHN_g$ is much faster than SA. In most cases, the $OSHN_g$ reached a final solution within a fraction of a second. The process then stopped and no further improvement could be made.

Although the SA approach is slow as compared to the $OSHN_g$, it is able to further improve the solution if a longer search time is allowed. For example, in comparison with the $OSHN_g$, the SA method has failed to obtain better solutions in four cases (problems 8 to 11), but it can reach a comparable solution if the maximum allowable number of iterations is increased. To illustrate, the SA algorithm has been repeated for the four problems with the maximum number of iterations set to 15000 instead of 2000. Table 4.3 shows the new η_g values for these problems.

No.	Problem Source	Size ($m \times n$)	Number of cells	$OSHN_g$ η_g (%)	SA η_g (%)	SA Iteration[a]
8	Chandrasekharan and Rajagopalan (1987)	40×100	10	94.81	94.41	12768
9	Chandrasekharan and Rajagopalan (1989, Fig. 1)	24×40	7	94.71	94.71	5249
10	Chandrasekharan and Rajagopalan (1989, Fig. 2.2)	24×40	7	93.56	93.56	5914
11	Chandrasekharan and Rajagopalan (1989, Fig. 2.3)	24×40	7	92.15	92.15	12229

[a] The iteration number at which the reported solution obtained

Table 4.3. The results obtained using the SA approach with 15000 iterations

It can be seen that after 15000 iterations, the SA approach obtained η_g values identical to those of the $OSHN_g$ in 3 cases (problems 9, 10 and 11). The η_g value for problem 8, the largest problem in the test set, is still slightly lower than that of $OSHN_g$ approach, which indicates longer search time is needed to further improve it.

In summary, our computations show that the SA approach can provide better solution quality though longer search time is needed. The striking feature of the $OSHN_g$, on the other hand, is its computational efficiency. Therefore, the two methods complement each other and provide alternative ways to solve comprehensive grouping problems.

CHAPTER 5

GROUP SCHEDULING

5.1 Background

The design issues in cellular manufacturing systems were addressed in Chapters 3 and 4. Once the machines and parts have been grouped, the remaining problem is how to sequence part families and schedule operations for the parts within each part family so that some planning goals can be achieved. This type of problem is called group scheduling and will be analyzed in this chapter. The maximization of machine utilization, workload balancing and minimization of job tardiness are among the most important objectives in group scheduling. The first two objectives have been considered in Chapters 3 and 4. The focus now is on the minimization of job tardiness.

The tardiness of each part depends on the completion time of the part which is a function of the machining speed of each operation of that part. Therefore, machining speed plays a key role in minimizing tardiness. Surprisingly, machining speed has not been considered in any group scheduling related study reported in the accessible literature. This is probably due to the increased complexity when machining speed is considered since group scheduling without machining speed change is already NP-hard. The schedule made without considering machining speed is no longer relevant to today's manufacturing environment where machining speed is easily controllable. To provide better solutions, machining speed is incorporated into the scheduling problem in this study.

The problem may be formally stated as follows. A CM system consists of a

number of machine cells. Each cell is mainly used by the part families that have been assigned to it in the design stage. However, parts from other cells are also allowed to be processed in this cell, and in this case they are treated as a separate part family for the concerned cell. Though parts in a family have similar operation requirements, their operation sequences could be quite different. To reduce setups, an exhaustive scheduling rule is applied, i.e., when a part family is being processed, it cannot be interrupted by a part from another family. Prior to processing each family, a major setup on each machine should take place, and for each single operation of a part a minor setup time is required. Each part has a pre-specified due date. If a part cannot be completed by its due date, a penalty cost will be considered. This penalty cost may be different for different parts. The objective is to find the best sequence for all part families and schedules for all parts of each family and to select optimal machining speed for each operation so that the total weighted tardiness is minimized.

5.1.1 Group scheduling model

The above specified group scheduling problem can be formulated as the following non-linear model:

Minimize

$$TT = \sum_{r=1}^{R} \sum_{j=1}^{N_r} \beta_{rj} T_{rj} \qquad (5.1)$$

subject to:

$$\sum_{r=1}^{R} y_{rf} = 1 \qquad \forall f \qquad (5.2)$$

$$\sum_{f=1}^{R} y_{rf} = 1 \qquad \forall r \qquad (5.3)$$

$$\sum_{j=1}^{N_r} x_{rijk} = 1 \qquad \forall r,i,k \tag{5.4}$$

$$\sum_{k=1}^{N_r} x_{rijk} = 1 \qquad \forall r,i,j \tag{5.5}$$

$$c_{r[i]j} - c_{r[i-1]j} \geq s_{r[i]j} + \frac{\lambda_{r[i]j}}{v_{r[i]j}} \qquad \forall r,j, \quad i=2, ..., M \tag{5.6}$$

$$c_{r[i]j} \geq \sum_{f=1}^{R} \left(\sum_{q=1}^{R} F_{q[i]}\, y_{q,f-1} \right) y_{rf} + p_r + \sum_{k=1}^{N_r} x_{r[i]jk} \sum_{l=1}^{k} \sum_{g=1}^{N_r} \left(s_{r[i]g} + \frac{\lambda_{r[i]g}}{v_{r[i]g}} \right) x_{r[i]gl} \quad \forall r,i,j \tag{5.7}$$

$$v_{rij}^{min} \leq v_{rij} \leq v_{rij}^{max} \qquad \forall r,i,j \tag{5.8}$$

$$y_{rf}\ ,\ x_{rijk} = 0 \text{ or } 1 \qquad \forall r,f,i,j,k \tag{5.9}$$

where

$$T_{rj} = \max\{0,\ c_{rj} - d_{rj}\} \tag{5.10}$$

$$c_{rj} = \max_{i} \{c_{rij} \mid i=1,...,M\} \tag{5.11}$$

$$F_{ri} = \max_{j} \{c_{rij} \mid j=1,...,N_r\} \tag{5.12}$$

The objective is to minimize the total weighted tardiness. Constraints 5.2 and 5.4 respectively specify that each part family appears in only one position of the family sequence and each part appears in only one position of the part sequence. Constraints 5.3 and 5.5 guarantee that each position of family or part sequences is occupied only once. Constraint 5.6 specifies the operation precedence for each part and ensures that the operation of a part on each machine cannot start unless the previous operation of the part is completed. Constraint 5.7 states that the process of a part on each machine cannot start until that machine becomes available. The last set of constraints specifies the range of

machining speed for each operation.

Because of the non-linearity and the large number of zero-one variables in the above model, the iterative search technique is adopted to solve this problem. In the following section, the solution procedure used in the iterative search process is explained.

5.2 Solution Procedure

5.2.1 Iterative search

In the iterative search process, three tasks are performed: a) machining speed selection, b) part family sequencing, and c) part scheduling (scheduling parts for each part family). Tasks (b) and (c) are performed in each iteration. The iterative search process begins with an initial feasible solution. During each iteration, a neighbourhood search is carried out. A neighbouring solution is obtained by interchanging positions of an existing part sequence in one of the part families without violating constraints. The objective value of the new schedule is then calculated according to Equation 5.1. The details of the three tasks to be performed are described below.

Machining speed selection

The guideline here is to specify a machining speed for each operation so that the total tardiness is minimized and production cost is reduced whenever possible. As mentioned earlier, the tardiness depends on the processing time, a function of the machining speed. Since the objective is to minimize the tardiness of each part, it is desirable to use the minimum processing time which is given by:

$$t_{rij}^{min} = \frac{\lambda_{rij}}{v_{rij}^{max}} \tag{5.13}$$

Therefore, the search should begin with the maximum speed. At the end of search, the slack times, if any, will be used to reduce the machining speed in order to minimize the production cost. A new issue to be addressed here is how much of the slack time should be used to reduce the production cost while the minimized total tardiness is maintained. This issue is addressed in the following.

The unit production cost, according to Hitomi (1979), is the sum of machining cost, material cost, tool cost, setup cost and overhead cost. For each operation, the speed that yields the minimum unit production cost is called the *minimum-cost speed* and denoted by v_{rij}^{c} as follows:

$$v^{c} = C^{T}\left[\frac{1}{\frac{1}{n^{T}}-1}\cdot\frac{k^{l}+k^{m}}{k^{l}t^{c}+k^{t}}\right]^{n^{T}} \tag{5.14}$$

Note that in the above equation and the discussions hereafter in this section subscripts *rij* are not included for simplicity. Figure 5.1 shows different situations of v^{c} with respect to the range of machining speed and the v^{slack} (the lowest possible cutting speed subject to the available slack time). Accordingly, the machining speed for each operation is determined as follows:

$$v = \begin{cases} \max(v^{c}, v^{slack}) & \text{if } v^{min} < v^{c} < v^{max} \\ v^{max} & \text{if } v^{c} > v^{max} \\ v^{slack} & \text{if } v^{c} < v^{min} \end{cases} \tag{5.15}$$

Part family sequencing

To sequence part families, the following concepts are introduced:

Part family makespan. This is the time interval from the time when the first operation of the part family starts to the time when the last operation of the family is completed.

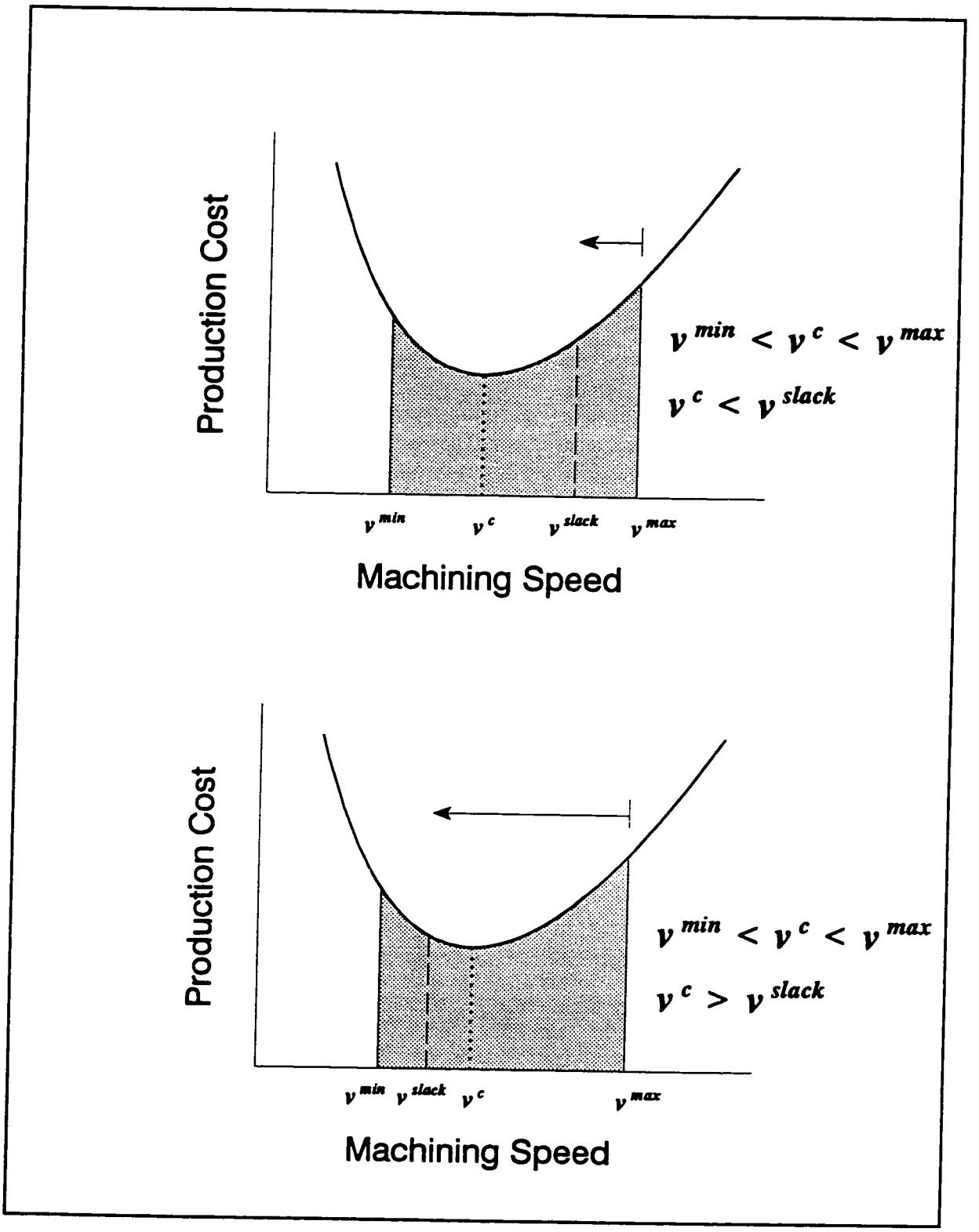


Figure 5.1(a) Unit production cost vs. machining speed, $v^{min} < v^c < v^{max}$

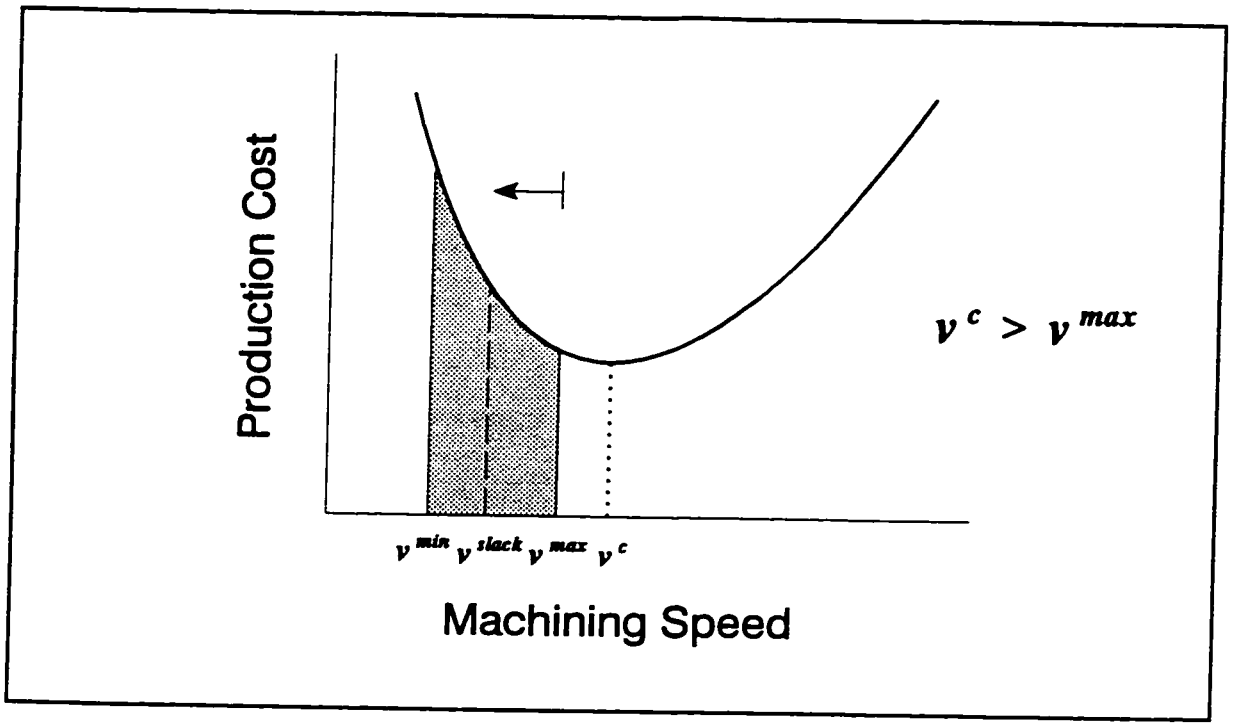


Figure 5.1(b) Unit production cost vs. machining speed, $v^c > v^{max}$

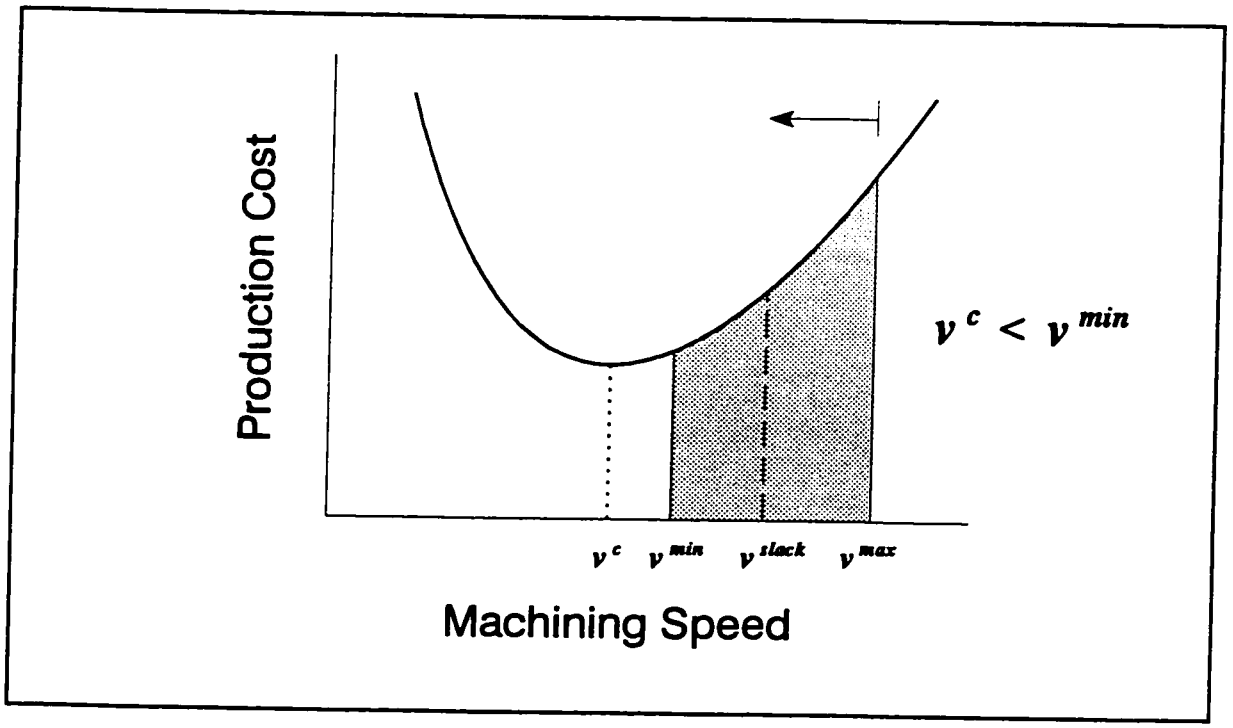


Figure 5.1(c) Unit production cost vs. machining speed, $v^c < v^{min}$

Average due date of part family. This is the arithmetic average of the due dates of all parts in a concerned part family.

Having defined the above concepts, the part family sequencing problem can be treated as a single machine sequencing problem and a proven efficient sequencing rule, MOD (modified operation due date) rule (Baker and Kanet 1983) can be applied. With the MOD rule, every part family will be scanned at a given time T against its modified due date, d_r^*, which is defined as:

$$d_r^* = \max \{\bar{d}_r , T+T_r\} \tag{5.16}$$

where $\bar{d}_r$ is the average due date of family r and is given by $(\Sigma_j d_{rj})/N_r$, and T_r is the makespan of family r. The family with the minimum d_r^* is processed next. Then this process repeats for the remaining families until all families have been sequenced in an iteration.

It is important to note that whenever the parts of a family are re-scheduled in the previous iteration, the makespan of that family, T_r, may also be changed. Therefore, the makespan of each part family has to be updated and all part families have to be re-scheduled using the MOD rule in each iteration.

Part scheduling

During each iteration, a new schedule or a neighbouring solution is obtained by switching the positions of two adjacent parts in a randomly selected part family. As mentioned earlier, this position switch may also change the makespan of the family. Therefore, the part families have to be re-scheduled using the MOD rule as described above.

Once the three tasks, i.e., machining speed selection, part family sequencing and part scheduling are completed a new iteration will start. This process continues until a specified termination condition is met. This process is summarized in Figure 5.2.

Figure 5.2 shows only the framework of the search, and other issues such as the neighbourhood search strategy, the criteria for accepting or rejecting a candidate solution and the mechanism of dealing with local optimum solutions are yet to be investigated.

In the next section, three search methods will be developed based on tabu search, simulated annealing and a new hybrid tabu-simulated annealing approach. The framework shown in Figure 5.2 will be used in all the three methods. The advantages and disadvantages of each method will be discussed and the performance of these methods will be compared.

5.2.2 Neighbourhood size

In all three methods, the search moves from current solution to a neighbouring solution. The selection of the neighbouring solution differs from method to method, but identical neighbourhood size is used in all three methods for later comparison purpose. In this study, a neighbouring solution can be generated by interchanging the positions of any two adjacent parts on a selected machine, and these two parts should belong to the same family. For example, suppose that a cell comprising of two machines, M_1 and M_2, is dedicated to two part families, $PF_1=\{1,2,3,4\}$ and $PF_2=\{5,6,7\}$. Assume that at a given iteration, the part family sequence of the current solution is (PF_1, PF_2) and the associated part sequences are:

$$PF_1 = \begin{Bmatrix} M_1: (3,2,4,1) \\ M_2: (2,1,3,4) \end{Bmatrix}, \quad PF_2 = \begin{Bmatrix} M_1: (6,7,5) \\ M_2: (5,6,7) \end{Bmatrix}$$

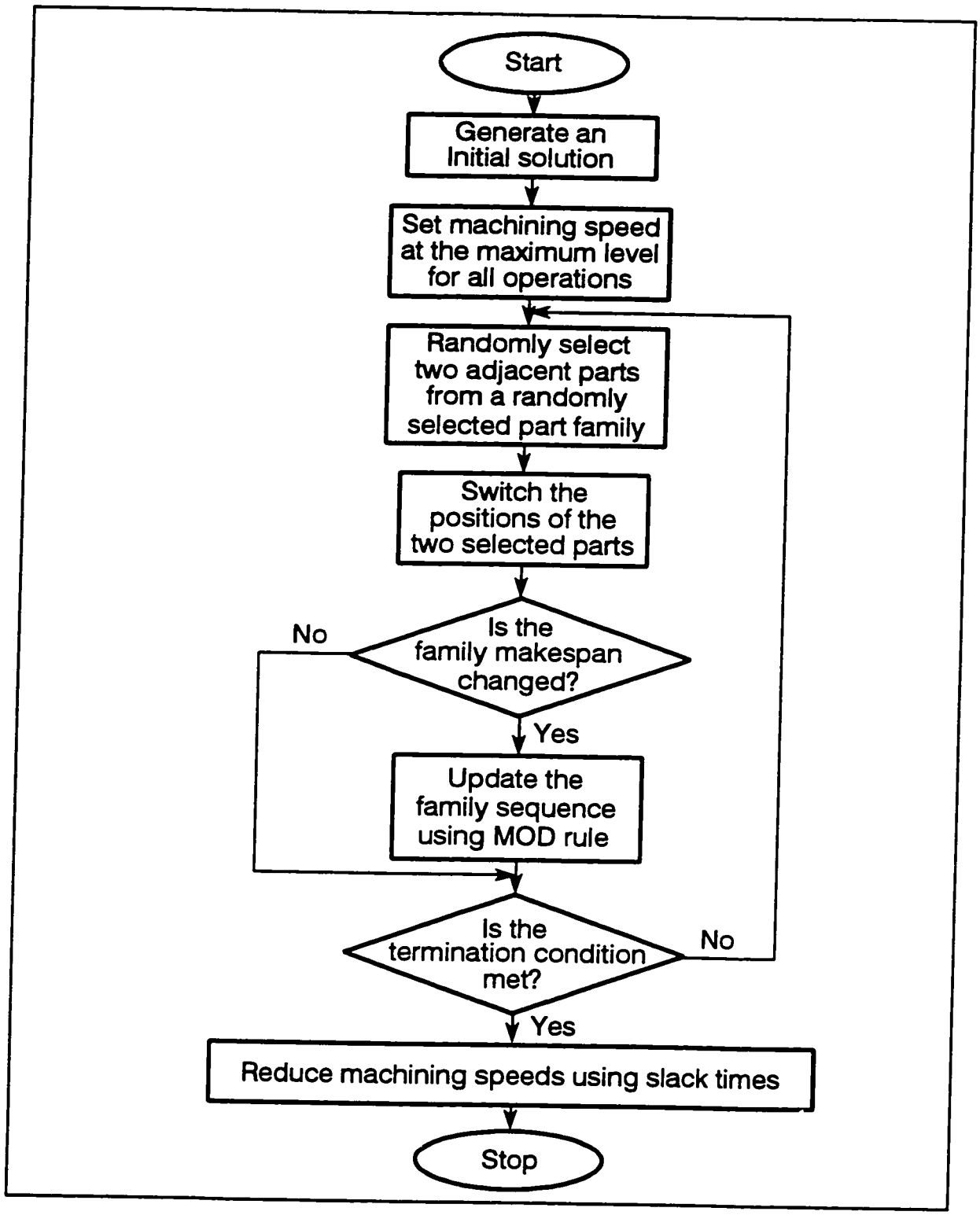


Figure 5.2. Framework of the iterative search process

The neighbourhood of this solution includes all the solutions that can be obtained by switching the positions of two adjacent parts. Some of these solutions are shown below. Note that bold numbers indicate the switched parts.

$$PF_1 = \begin{Bmatrix} M_1: (\mathbf{2},\mathbf{3},4,1) \\ M_2: (2,1,3,4) \end{Bmatrix}, \quad PF_2 = \begin{Bmatrix} M_1: (6,7,5) \\ M_2: (5,6,7) \end{Bmatrix}$$

$$PF_1 = \begin{Bmatrix} M_1: (3,\mathbf{4},\mathbf{2},1) \\ M_2: (2,1,3,4) \end{Bmatrix}, \quad PF_2 = \begin{Bmatrix} M_1: (6,7,5) \\ M_2: (5,6,7) \end{Bmatrix}$$

$$\vdots$$

$$PF_1 = \begin{Bmatrix} M_1: (3,2,4,1) \\ M_2: (2,1,3,4) \end{Bmatrix}, \quad PF_2 = \begin{Bmatrix} M_1: (6,7,5) \\ M_2: (5,\mathbf{7},\mathbf{6}) \end{Bmatrix}$$

In this case, for a group scheduling problem consisting of M machines and R part families, the neighbourhood size, *NSIZE*, is calculated as follows:

$$NSIZE = \sum_{r=1}^{R} M(N_r - 1) \tag{5.17}$$

where N_r represents the number of parts in family r. For the above example, the neighbourhood size is 10.

5.3 Iterative Search Methods

5.3.1 Tabu search

Tabu search, originally proposed by Glover (1977), is a deterministic search technique which is able to escape local optima by using a list of prohibited neighbouring solutions known as *tabu list*. This search technique has been successfully applied to many combinatorial optimization problems. The tabu search process starts with a feasible

solution, and during each iteration evaluates all solutions in the neighbourhood except those in the current tabu list, i.e., a list of several immediate previous moves or solutions. Then the search moves to the best non-tabu solution in the neighbourhood and the best solution found so far is updated if necessary. When the tabu list is full and a new entry arrives, the oldest entry in the list will be popped up and removed from the list. In addition to escaping local optima, using the tabu list can also avoid re-visiting the recent neighbours recorded in the list and thus save computational time.

Tabu search has two other features that make it more sophisticated, *aspiration* and *diversification*. Aspiration is a criterion that allows the search to override the tabu status of a solution. This feature provides backtracking to recent solutions if they can lead to a new path to better solutions. Furthermore, aspiration prevents the search from being trapped into a solution surrounded by tabu neighbours. This is more likely to occur when the size of neighbourhood is less than or equal to the size of tabu list. Figure 5.3 illustrates such a case where the number in a block is the objective value associated with a feasible solution. Suppose the objective value is to be minimized, the tabu-list size is set to 8, and the neighbourhood size is 4 (Only the rectilinear moves are allowed). When the search moves to the solution with cost of 8 (i.e., the central block) following the search path as shown, all of the neighbouring solutions are tabu, i.e., all the 4 surrounding solutions have been included in the current tabu list. Therefore, the search will freeze unless at least one neighbour's tabu status can be ignored by an aspiration criterion. For example, the aspiration criterion could be satisfied if all the solutions in the current neighbourhood are tabu, i.e., all of them are in the tabu list. In this case, the tabu status of the best neighbouring solution (i.e., the solution with cost of 4) can be ignored and the move will be made to this solution.

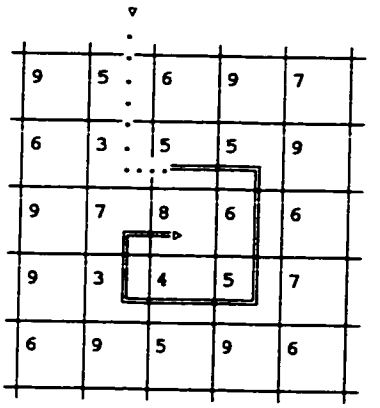


Figure 5.3. A tabu search example with tabu length greater than neighbourhood size
(.... search path , = current tabu list)

Diversification is often used to explore some sub-domains that may not be reached otherwise. Diversification is done by redirecting the search path or re-starting the search from a different initial solution. Without diversification, the search may be limited to a subset of the solution space. This can be illustrated in Figure 5.4. In this example, the size of the tabu list is 5, the neighbourhood size is 4, and only rectilinear moves are allowed. To simplify discussions, a new concept, *partition set*, is introduced. A partition set is defined as a set of connected neighbours separating the entire solution space into two sections and having objective values higher than those of adjacent neighbours in the two separated sections. In Figure 5.4, three partition sets are shown in shade. Note that in reality solutions in a partition set may have different cost values, but in this example equal values are assigned for simplicity. A partition set could be either closed or open. In Figure 5.4, partition sets with the cost of 8 and 9 are closed sets, and the one with the cost of 7 is an open one.

As shown in Figure 5.4, any of the three partition sets will divide the entire domain into two sections. Without diversification, the search will not be able to pass the

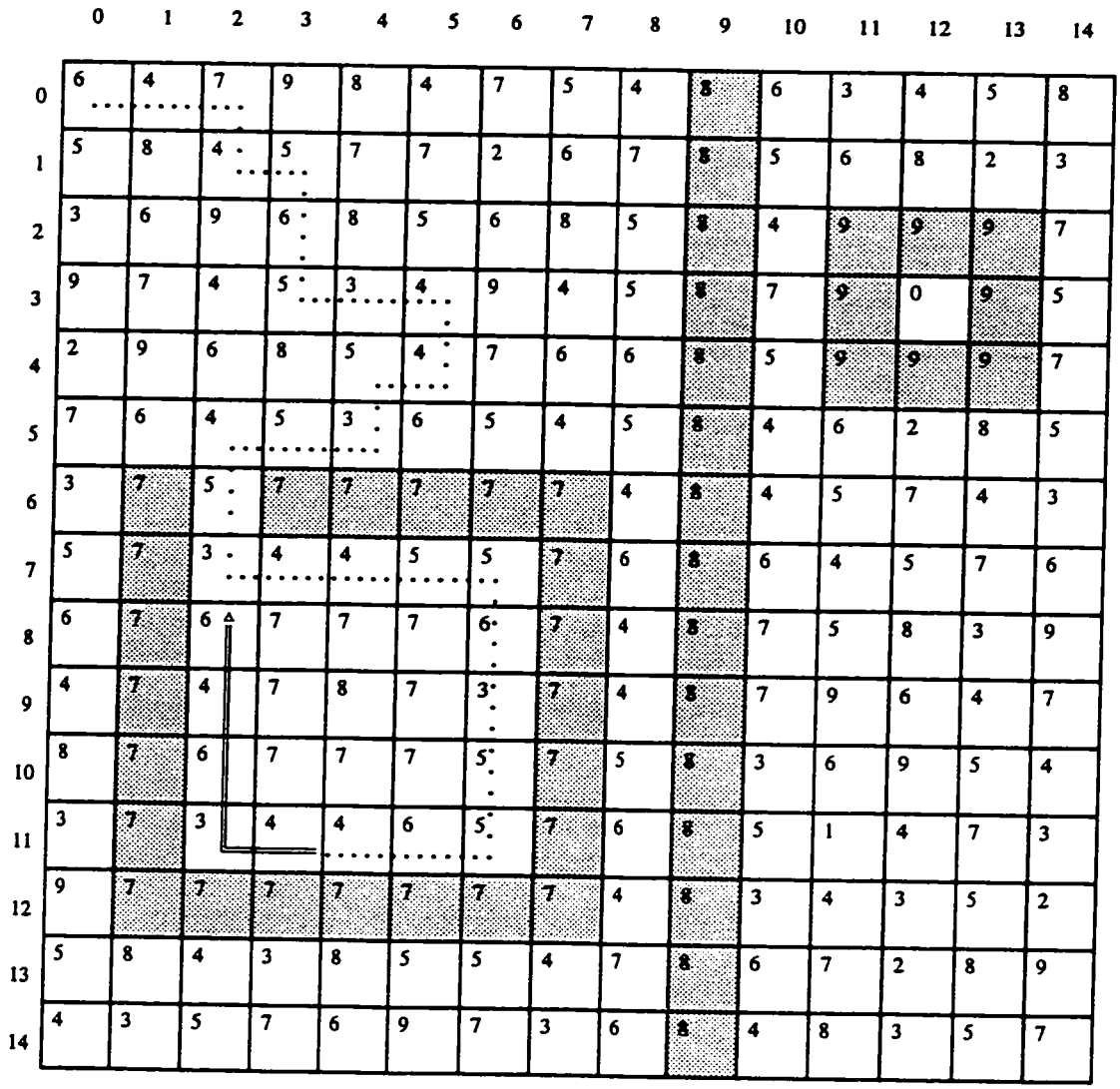


Figure 5.4. An example of a solution space divided into subsets
(.... search path , == current tabu list)

partition barriers. As a result, two very undesirable situations may arise: a) the search may be trapped to a very small area such as the one surrounded by 7 in Figure 5.4 and thus the global optimum may never be located; and b) the search may be performed over most of the area but may never find the global optimum simply because it is enclosed by

a partition "wall" such as the partition set with value 9 in Figure 5.4 (in this case, it is assumed that the other two partition sets do not exist). Diversification makes it possible to restart the search directly from a new subset by simply re-locating the next search point to that subset. In Figure 5.4, the optimum solution with the cost of 0 cannot be reached unless the search starts from one of its immediate neighbours, and this is possible only when a diversification strategy is used. It should be pointed out that if there are too many partition sets in the solution space, the commonly used diversification methods such as re-starting from the best solution obtained so far and re-starting from a randomly selected point may not work.

To solve the group scheduling problem, a tabu search algorithm incorporating both aspiration and diversification strategies has been developed and is presented as follows.

Algorithm 5.1

Step 1 Read *Tabu_length*, *Move_max*, and *Dive_max*, build *Tabu_list* with length of *Tabu_length*, set $t \leftarrow 0$, $d \leftarrow 0$ and set the aspiration criterion (the aspiration criterion is satisfied if the entire neighbourhood is tabu).

Step 2 *Machining speed selection*

For all operations set $v_{rij} \leftarrow v_{rij}^{max}$.

Step 3 *Initialization*

1) Set an initial solution, $S_t \leftarrow S_0$, by generating a feasible schedule.

2) Calculate the total weighted tardiness for the initial solution, $TT(S_t)$; and set $TT(Solution\_best) \leftarrow TT(S_t)$.

Step 4 *Neighbourhood search*

1) Set *TT(Neighbour_best)* to a large number.

2) If the entire neighbourhood has been searched, go to Step 7; otherwise go to the next step.

3) Generate a neighbouring solution from S_t, by switching two adjacent parts in a family and updating the family sequence using the MOD rule.

4) Calculate the total weighted tardiness for the neighbouring solution, *TT(Neighbour)*.

5) If *TT(Neighbour)* $\geq$ *TT(Neighbour_best)*, discard this neighbouring solution and go to (2) of Step 4; otherwise, continue.

6) If the neighbouring solution is in *Tabu_list*, continue; otherwise go to Step 6.

Step 5 *Aspiration*

If the neighbouring solution satisfies the aspiration criterion, go to the next step; otherwise, go to (2) of Step 4.

Step 6 Set *TT(Neighbour_best)*$\leftarrow$*TT(Neighbour)*, and go to (2) of Step 4.

Step 7 Set $t \leftarrow t+1$, $d \leftarrow d+1$, and accept the *Neighbour_best* solution as the next iteration solution (i.e., $S_t \leftarrow$ *Neighbour_best*).

Step 8 Update *Tabu_list*.

Step 9 If *TT*(S_t) < *TT(Solution_best)*, set *TT(Solution_best)*$\leftarrow$*TT*(S_t); otherwise, go to the next step.

Step 10 *Diversification*

If d=*Dive_max*, set $d \leftarrow 0$, and go to Step 3; otherwise, go to the next step.

Step 11 *Termination*

If t < *Move_max*, go to Step 4; otherwise, go to the next step.

Step 12 *Machining speed adjustment:*

For all operations set the machining speed using Equation 5.15, stop.

A drawback of tabu search is that if it reaches a previously visited solution, it will cycle following the same path unless a tabu neighbour exists. In other words, if the search moves to a previously visited solution that has not been tabu for the last two iterations, then a loop is encountered. Such a case is shown in Figure 5.4. Starting from the solution located at (0,0), the search follows the path as shown till it reaches the solution at (8,2) with the cost of 6. At this point, a move is made to the previously visited solution at (7,2), and cycling starts along the inner wall of the partition barrier. To alleviate this difficulty, a longer tabu list or larger neighbourhood size may be used, but this may cause longer computational time. Therefore, cycling may not be avoided when tabu search algorithm is applied and a better diversification strategy is needed.

5.3.2 Simulated annealing

The cycling problem present in tabu search can be avoided using simulated annealing (SA) method. Furthermore, SA does not need to evaluate the entire neighbourhood in order to make a move, and thus needs less computational time at each iteration. For these reasons, an SA algorithm will be developed as an alternative to solve the group scheduling problem in this study. Simulated annealing, as discussed in Chapter 4, is a random search technique that is able to escape local optima using a probability function. Unlike tabu search, SA does not evaluate the entire neighbourhood at each iteration. Instead, it randomly chooses only one solution from the current neighbourhood and evaluates its cost. If the cost is improving, it makes the move; otherwise, the move

is made only if the transition probability is higher than a uniform random number. The transition probability depends on the difference between costs of the current solution and the neighbouring solution. If the transition from the current solution to the candidate neighbour is rejected, then another solution in the neighbourhood will be selected and evaluated. The transition probability is as follows:

$$Pr\left(Solution\_current_{t+1} = Solution\_neighbor_t\right) = min\left\{1, \exp\left(-\frac{\Delta C}{\theta_t}\right)\right\} \qquad (5.18)$$

where ΔC is the cost difference between the current and the neighbouring solutions, t is the iteration number, and θ_t is a control parameter known as temperature. The search initially starts from a high temperature with a better chance of transition from a low cost solution to a high cost solution (i.e., escaping local minima). By reducing the temperature while the iteration number is increasing, the transition probability declines toward zero. When the temperature approaches zero, the search is no longer able to escape local minima, and a move is made only if the cost is improving. In this thesis the temperature declining is achieved using the cooling schedule given by Equation 4.8.

Examining the above probability function shows that both the cost difference, ΔC, and the temperature, θ_t, contribute to the possibility of transition to a higher cost. When the temperature reduces, the transition probability is decreased. This probability is however lower when ΔC is larger or temperature is lower.

The SA algorithm for the group scheduling problem is summarized as follows.

Algorithm 5.2

Step 1 Read θ_0 and *Move_max*, and set $t \leftarrow 0$.

Step 2 *Machining speed selection*

For all operations set $v_{rij} \leftarrow v_{rij}^{max}$.

Step 3 *Initialization*

1) Set an initial solution, $S_t \leftarrow S_0$, by generating a feasible schedule within each family and obtaining family sequence using the MOD rule.

2) Calculate the total weighted tardiness for the initial solution, $TT(S_t)$; and set $TT(Solution\_best) \leftarrow TT(S_t)$.

Step 4 *Neighbouring solution*

1) Generate a neighbouring solution, $Solution\_neighbour_t$ from the current solution S_t by switching two randomly selected adjacent parts in a family and updating the family sequence using the MOD rule.

2) Calculate the total weighted tardiness for the neighbouring solution, $TT(Solution\_neighbour_t)$.

3) If $TT(Solution\_neighbour_t) < TT(S_t)$, go to Step 6; otherwise, continue.

Step 5 *Transition probability*

1) Update the temperature according to Equation 4.8 and calculate the transition probability, $Pr(S_{t+1} = Solution\_neighbour_t)$, using Equation 5.16.

2) Generate a uniform random number, α, from the interval [0,1].

3) If $Pr(S_{t+1} = Solution\_neighbour_t) < \alpha$, go to Step 7; otherwise, go to the next step.

Step 6 Accept the neighbouring solution as the next iteration solution (i.e., $S_{t+1} \leftarrow Solution\_neighbour_t$).

Step 7 Set $t \leftarrow t+1$.

Step 8 If $TT(S_j) < TT(Solution\_best)$, set $TT(Solution\_best) \leftarrow TT(S_j)$; otherwise, go to the next step.

Step 9 *Termination*

If $t < Move\_max$, go to Step 4; otherwise, go to the next step.

Step 10 *Machining speed adjustment*

For all operations set the machining speed using Equation 5.15, stop.

Though the SA algorithm has some advantages, it needs more iterations to find the best solution. Another drawback of SA as compared to tabu search is that it does not have memory and hence is possible to return to some recently visited solutions. This often leads to more iterations and longer computational time. The limitations of tabu search and SA algorithms give rise to a new approach which take advantage of both tabu search and SA. The details of the new approach are explained in the next section.

5.3.3 Tabu-simulated annealing

As discussed above, tabu search uses the short-term memory (tabu list) to escape local optima, but it cannot avoid cycling because of its deterministic nature. Simulated annealing, on the other hand, can avoid cycling due to its stochastic characteristic but suffers from slow rate of improvement. One reason for this slow improvement lies in the fact that simulated annealing does not keep track of previously visited solutions, and it is always possible for the search to return to a recently visited solution. If the search is prohibited from returning to these previous solutions by using the short term memory, the performance of the simulated annealing approach can be significantly improved. Based on this idea, an SA approach supplemented with a tabu list, called *tabu-simulated*

annealing (tabu-SA) is proposed in this thesis. This hybrid method is expected to reduce the number of revisits to previous solutions and cover a broader range of solutions.

The tabu-SA algorithm for the group scheduling problem is given below.

Algorithm 5.3

Step 1 Read *Tabu_length*, *Move_max* and θ_0, build *Tabu_list* with length of *Tabu_length*, and set $t \leftarrow 0$, $d \leftarrow 0$, and set the aspiration criterion (the aspiration criterion is satisfied if the entire neighbourhood is tabu).

Step 2 *Machining speed selection:*

For all operations set $v_{rij} \leftarrow v_{rij}^{max}$.

Step 3 *Initialization:*

1) Set an initial solution, $S_t \leftarrow S_0$, by generating a feasible schedule within each family and obtaining family sequence using the MOD rule.

2) Calculate the total weighted tardiness for the initial solution, $TT(S_t)$; and set $TT(Solution\_best) \leftarrow TT(S_t)$.

Step 4 *Neighbouring solution:*

1) Generate a neighbouring solution, $Solution\_neighbour_t$ from the current solution S_t by switching two randomly selected adjacent parts in a family and updating the family sequence using the MOD rule.

2) If $Solution\_neighbour_t$ is in *Tabu_list*, go to the next step; otherwise go to Step 6.

Step 5 *Aspiration:*

If the neighbouring solution satisfies the aspiration criterion, go to

the next step; otherwise, go to Step 4.

Step 6 Calculate the total weighted tardiness for *Solution_neighbour*, *TT(Solution_neighbour$_j$)*.

Step 7 If *TT(Solution_neighbour$_j$)* < *TT(S$_j$)*, go to Step 9; otherwise, go to the next step.

Step 8 *Transition probability:*

1) Update the temperature according to Equation 4.8 and calculate the transition probability, $Pr(S_{t+1}=Solution\_neighbour_j)$, using Equation 5.16.

2) Generate a uniform random number, α, from the interval [0,1].

3) If $Pr(S_{t+1}=Solution\_neighbour_j) < \alpha$, go to Step 11; otherwise, go to the next step.

Step 9 Accept the neighbouring solution as the next iteration solution (i.e., $S_{t+1} \leftarrow Solution\_neighbour_j$).

Step 10 Update *Tabu_list*.

Step 11 Set $t \leftarrow t+1$.

Step 12 If *TT(S$_j$)* < *TT(Solution_best)*, set *TT(Solution_best)*←*TT(S$_j$)*; otherwise, go to the next step.

Step 13 *Termination:*

If $t < Move\_max$, go to Step 4; otherwise, go to the next step.

Step 14 *Machining speed adjustment:*

For all operations set the machining speed using Equation 5.15, stop.

5.4 Implementation of Heuristics

5.4.1 Illustrative examples

The above three algorithms have been coded in C and tested using two example problems. Ten runs are carried out for each example and each algorithm. Each run starts with a distinct seed solution. For comparison purpose, the same set of seed solutions are used for every method. For Algorithms 5.1 and 5.3 the aspiration level is reached if the entire neighbourhood is tabu. Diversification in Algorithm 5.1 is considered only when a certain number of iterations has been completed. To diversify the search, a new initial solution is obtained by switching the first part and the last part in the largest part family.

Example 1

The data for the first example problem is provided in Table 5.1. The tabu length in algorithms 5.1 and 5.3 was set to 7. The total number of iterations in each run was 100. The results are reported in Table 5.2.

Family r	Part j	Machine i	Operation order	s_{rj}	t_{rj}^{min}	t_{rj}^{max}	d_{rj}	p_r
1	1	1	2	2	1	3	13	1
		2	1	1	2	5		
		3	3	1	1	4		
	2	1	2	1	1	4	10	
		2	3	1	2	4		
		3	1	2	2	3		
	3	1	1	1	1	6	12	
		2	2	1	2	3		
		3	3	1	1	3		
2	1	1	1	1	2	4	15	2
		2	2	1	1	5		
		3	-	-	-	-		
	2	1	2	1	2	3	18	
		2	-	-	-	-		
		3	1	1	1	4		

Table 5.1. Input data for Example 1

The results indicate that the solutions obtained by tabu search are worse than those of the other two algorithms in most cases. Such a poor performance is probably caused by the fixed neighbourhood size. A dynamic neighbourhood size proposed by Adenso-Díaz (1992), which begins with a large neighbourhood and reduces the neighbourhood size as the search advances, could lead to a better performance. In this study, however, in order to have a fair comparison among the three heuristics, a fixed neighbourhood size is used.

It can be seen that the tabu-simulated annealing approach has outperformed the other two in terms of solution quality. The hybrid approach has obtained the lowest total tardiness for all ten runs. Though the SA method reached to the lowest tardiness in eight out of ten runs, the other two runs give unacceptably high tardiness. The computational time for the hybrid approach is slightly higher than that of SA but significantly shorter than the tabu search method. The best solution obtained for Example 1 is depicted in Figure 5.5.

Run No.	Initial total weighted tardiness	Final total weighted tardiness		
		Tabu	SA	Tabu-SA
1	55	46	3	3
2	52	34	3	3
3	40	5	3	3
4	43	40	3	3
5	48	34	3	3
6	31	3	9	3
7	49	5	3	3
8	34	31	3	3
9	39	36	24	3
10	25	5	3	3
Average computational time per run (sec)		0.60	0.18	0.20

Table 5.2. Final results for Example 1 (100 iterations for each run)

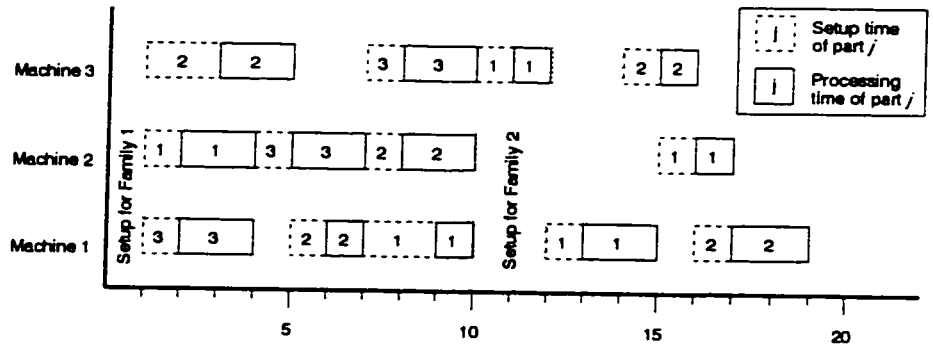


Figure 5.5. The best solution obtained for Example 1 after 100 iterations

Example 2

A larger problem is considered in this example. The data for this problem are listed in Table 5.3. The tabu length in algorithms 5.1 and 5.3 was set to 10 and the number of iterations in each of the ten runs was set to 1000. Table 5.4 summarizes the computational results. The tabu-SA approach again provides better solution quality than the other two algorithms. With the tabu-SA method, the lowest tardiness level, 162, has been reached four times, and all the objective values are less than or equal to 171. The SA algorithm, however, reached to the lowest value of 162 only once. Figure 5.6 shows the convergence process during each run for the three heuristics.

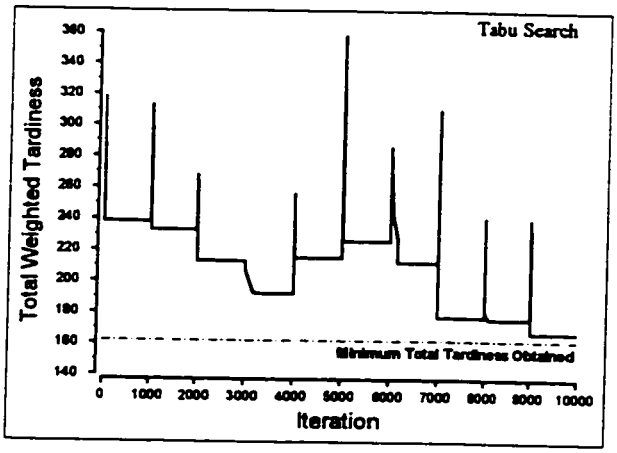
The computation reveals that for the first example almost 48% of neighbouring solutions were tabu and thus were not evaluated by the tabu-SA approach. This value for the second example was about 24%. These are the percentages of revisited solutions that has been avoided by the tabu-SA approach. The magnitude of this value depends on the relative size of tabu length and the neighbourhood size. The computations also show that if the tabu length is relatively high comparing to the neighbourhood size (e.g., in the first example), then the percentage of revisits during the search will be low and the percentage is high otherwise.

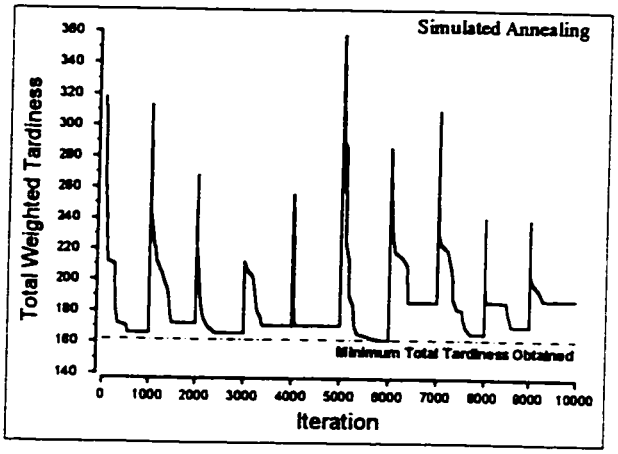
Family r	Part j	Machine i	Operation order	s_{rj}	t_{rij}^{min}	t_{rij}^{max}	d_{rj}	p_r
1	1	1	1	3	11	18	145	25
		2	4	1	1	1		
		3	-	-	-	-		
		4	2	1	1	2		
		5	3	1	1	2		
	2	1	1	3	15	27	129	
		2	5	1	1	1		
		3	4	2	5	8		
		4	2	1	1	2		
		5	3	1	1	2		
	3	1	1	2	13	21	164	
		2	-	-	-	-		
		3	4	2	4	8		
		4	2	1	1	2		
		5	3	1	1	2		
	4	1	1	1	8	13	91	
		2	-	-	-	-		
		3	2	3	18	35		
		4	3	1	2	4		
		5	4	1	1	3		
2	1	1	1	2	7	12	59	16
		2	2	1	1	1		
		3	-	-	-	-		
		4	3	1	1	2		
		5	-	-	-	-		
	2	1	1	3	10	15	103	
		2	3	1	2	2		
		3	-	-	-	-		
		4	2	1	2	3		
		5	-	-	-	-		
	3	1	1	2	11	20	88	
		2	3	1	1	1		
		3	-	-	-	-		
		4	2	1	2	4		
		5	-	-	-	-		
3	1	1	1	1	6	11	112	12
		2	-	-	-	-		
		3	2	2	13	27		
		4	-	-	-	-		
		5	-	-	-	-		
	2	1	1	2	7	11	86	
		2	-	-	-	-		
		3	2	2	9	18		
		4	-	-	-	-		
		5	-	-	-	-		

Table 5.3. Input data for Example 2

Run No.	Initial total weighted tardiness	Final total weighted tardiness		
		Tabu	SA	Tabu-SA
1	318	238	166	167
2	313	233	172	162
3	268	213	166	162
4	212	192	171	166
5	256	215	171	162
6	358	226	162	171
7	286	212	186	171
8	310	177	166	171
9	241	176	171	162
10	240	167	188	171
Average computational time per run (sec)		103.86	5.79	6.25

Table 5.4. Final results for Example 2 (1000 iterations for each run)





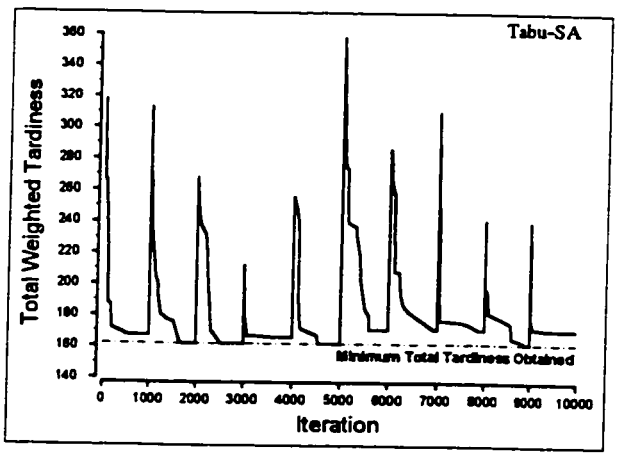


Figure 5.6. Comparison of convergence processes during 10 runs

5.4.2 Test problems

The above illustrative examples show that the solutions obtained using tabu-SA approach are more consistent and favourably comparable with the other two heuristics. To further investigate the performance of these three heuristics, 10 additional test problems have been generated and solved using the three heuristics. Similar to the above illustrative examples, for each problem and each algorithm 10 runs are carried out and each run starts with a particular seed solution which is identical for all three approaches. Since the time required for each iteration is quite different for the three algorithms, total search time instead of maximum number of iterations is used as the termination criterion. The allowable search time for each run is set to 10 seconds for the first 4 problems and 30 seconds for the rest. The computational results are summarized in Table 5.5.

In Table 5.5, the three approaches are compared against: 1) the minimum total tardiness, 2) the maximum total tardiness, 3) the average total tardiness, and 4) the standard deviation of the total tardiness values. These statistics are calculated based on the 10-run results of each test problem.

Table 5.5 clearly shows that, for the same amount of search time, the tabu-SA approach provides better or at least equal solutions (the standard deviation of test problem number 1 and the minimum tardiness of test problem number 2) for all cases with only one exception, the standard deviation of test problem number 7 where the value is slightly higher than the one obtained using the SA method. To further examine the performance of three methods, the convergence processes are depicted in Figure 5.7 for all the test problems.

As can be seen in Figure 5.7, of the three methods the tabu search approach has the slowest convergence rate. In many cases, the SA method provides fast convergence rate in the first few seconds of the search process but then slows down or freezes at a

Problem No.	Number of Machines	Number of Families	No. of Parts in Each Family	Approach	Total Weighted Tardiness			
					Min [a]	Max [b]	Avg [c]	SD [d]
1	10	3	{4,5,3}	Tabu	138	343	233	77
				SA	123	167	135	14
				Tabu-SA	120	159	133	14
2	15	2	{12,8}	Tabu	225	1031	445	271
				SA	0	126	26	39
				Tabu-SA	0	82	21	33
3	4	4	{5,7,4,8}	Tabu	307	571	427	85
				SA	151	207	171	15
				Tabu-SA	143	171	162	9
4	7	2	{11,8}	Tabu	2580	4917	3315	690
				SA	475	693	536	62
				Tabu-SA	436	580	520	44
5	6	2	{18,15}	Tabu	3533	9999	6416	2618
				SA	109	1263	548	472
				Tabu-SA	104	1170	390	415
6	9	4	{7,4,5,2}	Tabu	1420	3124	2295	614
				SA	1337	1677	1447	131
				Tabu-SA	1315	1542	1414	83
7	12	3	{7,3,5}	Tabu	1314	2869	1993	537
				SA	872	1209	990	111
				Tabu-SA	849	1194	974	115
8	20	2	{14,11}	Tabu	510	1754	953	348
				SA	218	634	384	131
				Tabu-SA	215	429	358	71
9	14	3	{10,6,5}	Tabu	1716	2581	2184	263
				SA	205	1179	551	380
				Tabu-SA	196	1010	377	260
10	11	3	{12,8,5}	Tabu	2995	5981	4648	1058
				SA	126	399	216	86
				Tabu-SA	97	357	204	80

[a] The minimum of the 10-run results
[b] The maximum of the 10-run results
[c] The average of the 10-run results
[d] The standard deviation of the 10-run results

Table 5.5. Summary of the results of 10 test problems

certain level. The tabu-SA method, on the other hand, displays a quite robust convergence behaviour. The possible reason is that, as observed in section 5.4.1, a significant number of "revisits" will occur if a tabu list is not used (e.g., 48% for Example 1 and 24% for Example 2). This type of revisits can be significantly reduced by using the tabu-SA method, thus rendering a more robust search process.

5.5 Discussion

From the above computational results, the following advantages of the tabu-SA method can be observed.

(1) Insensitive to partition sets

Due to the stochastic nature of the tabu-SA approach, the search path is insensitive to and hence not restricted by the partition sets as discussed in section 5.3. Thus, one of the main difficulties encountered in tabu search has been avoided and consequently the diversification strategy used in the tabu search method become unnecessary.

(2) Avoided cycling

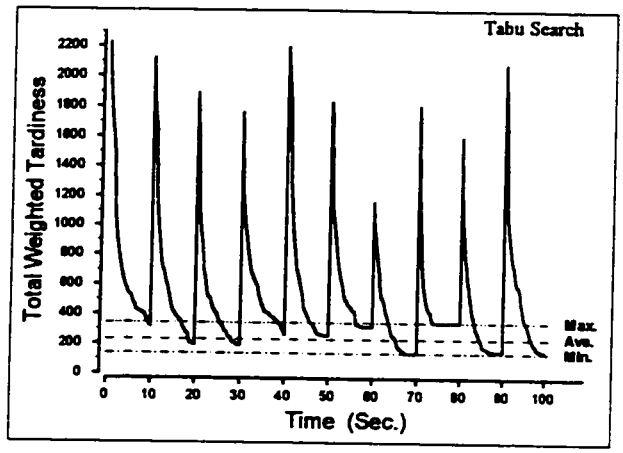
Cycling has been a major hurdle of tabu search method which traps the search process into local optimum. The deterministic process of the tabu search is generally considered to be the main reason of cycling. The probabilistic search path of the new tabu-SA method can effectively avoid cycling.

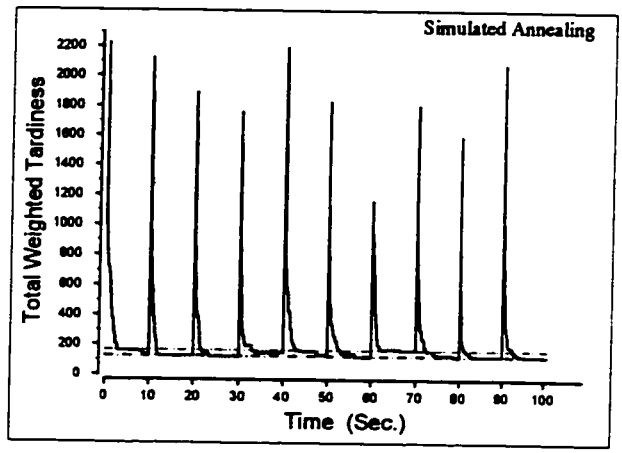
(3) Less revisits

Due to the short term memory feature added to the tabu-SA method, the number of solution revisits can be significantly reduced comparing to the SA algorithm. As a

result, the computational time can be reduced considerably.

All of the above will positively contribute to the solution quality. This has been demonstrated by the computational results as summarized in Table 5.5 and Figure 5.7. It is also important to note that the tabu-SA approach yields more consistent solutions as indicated by the standard deviation values of the 10-run solutions (Table 5.5).





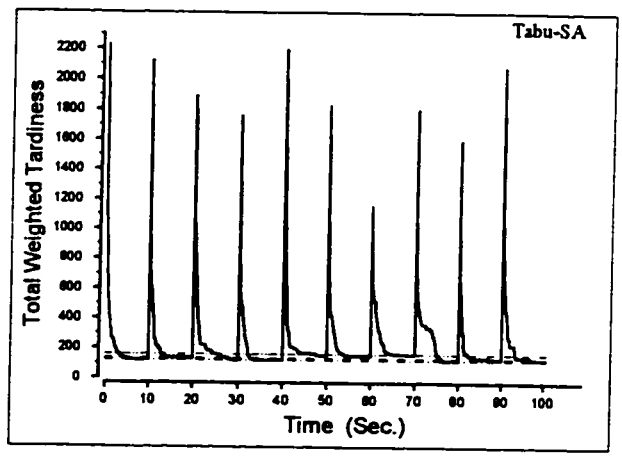
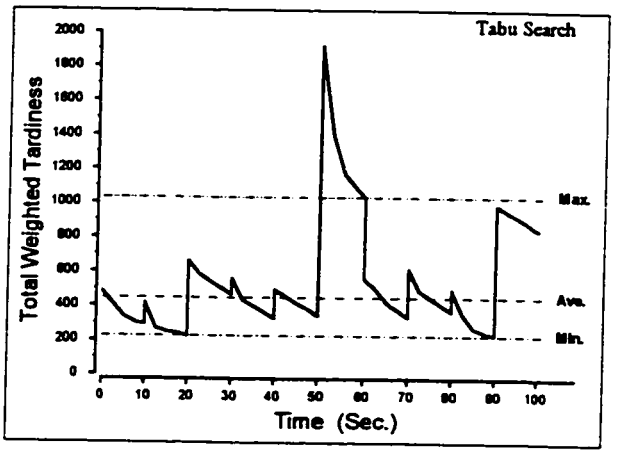
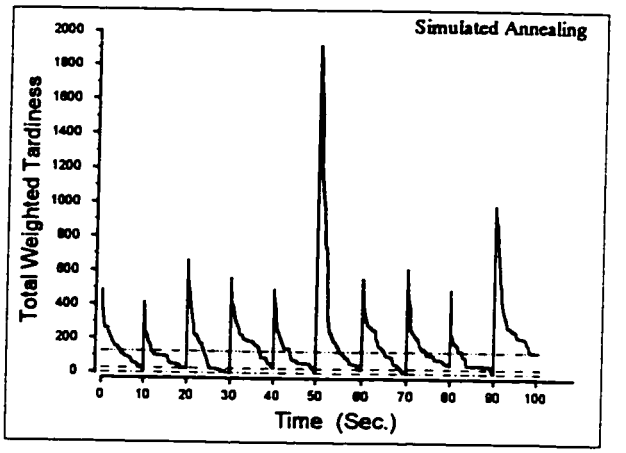


Figure 5.7(a) Comparison of convergence processes for test problem 1





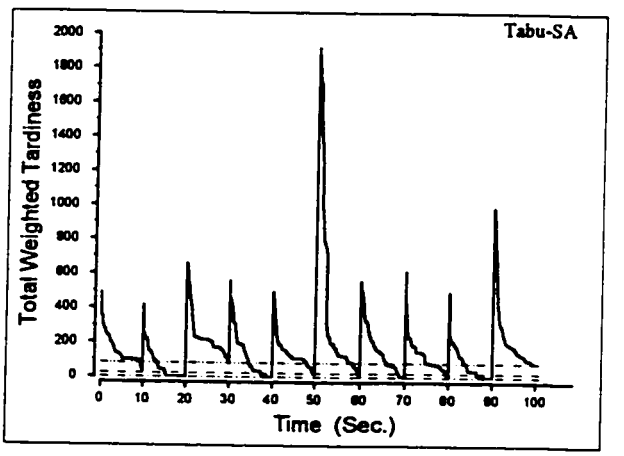
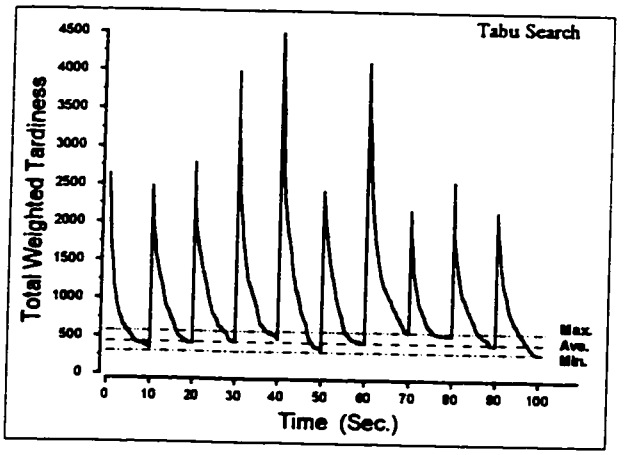
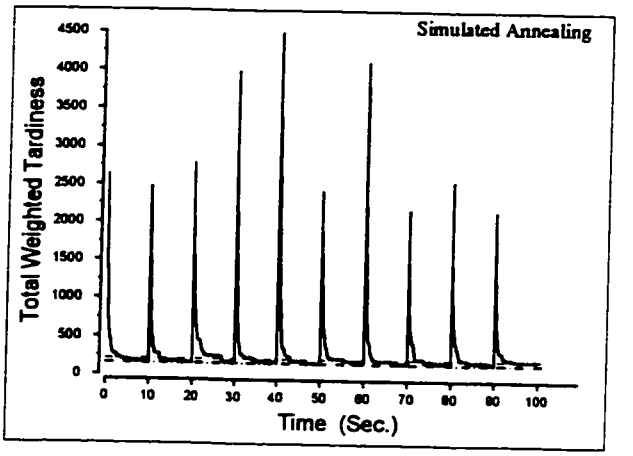


Figure 5.7(b) Comparison of convergence processes for test problem 2





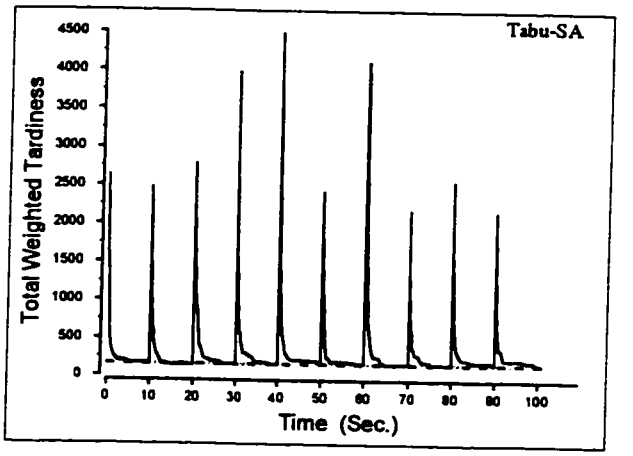
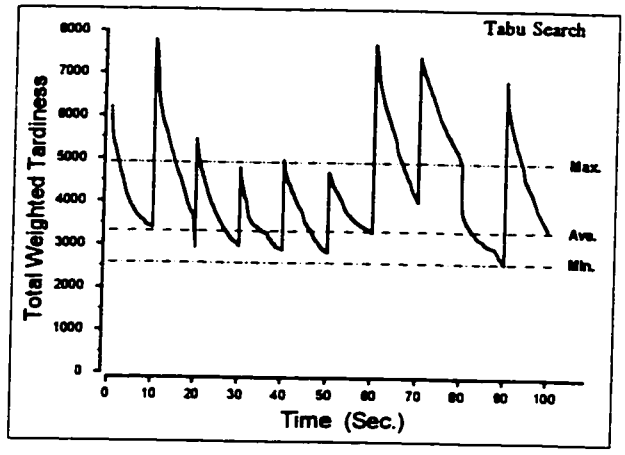
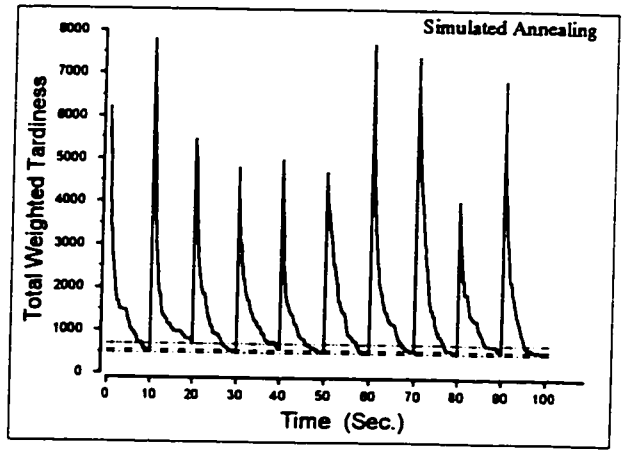


Figure 5.7(c) Comparison of convergence processes for test problem 3





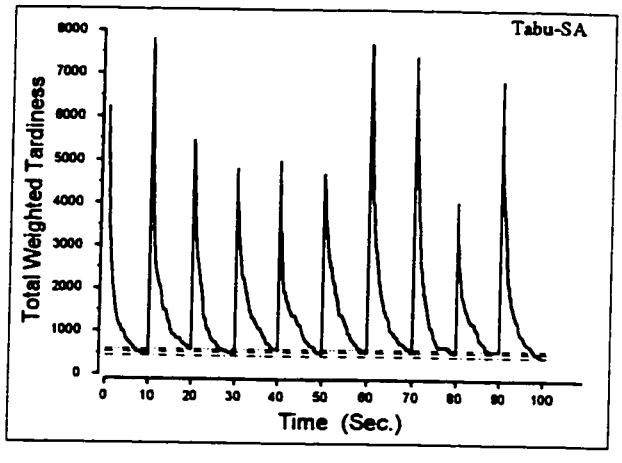
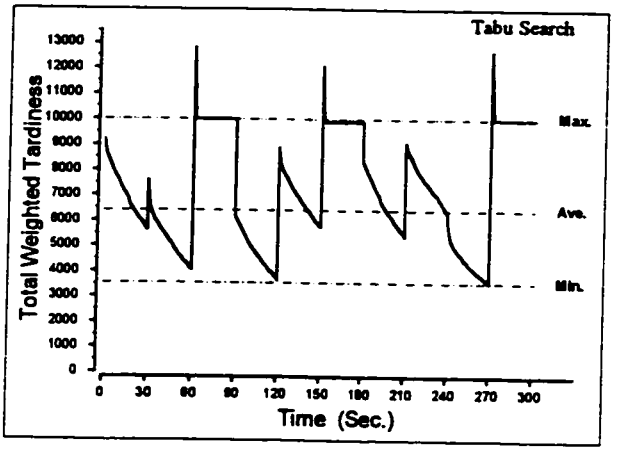
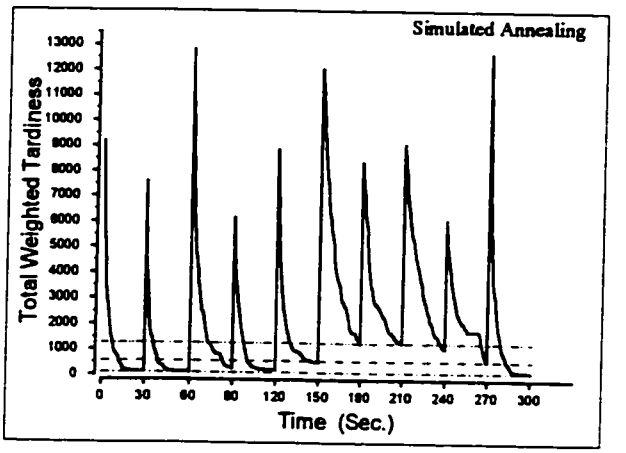


Figure 5.7(d) Comparison of convergence processes for test problem 4





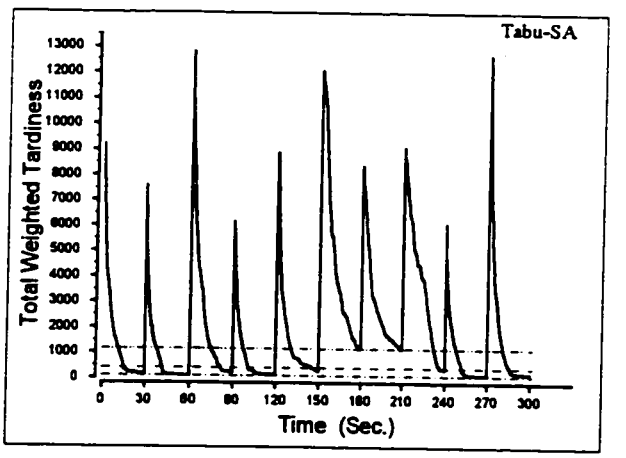
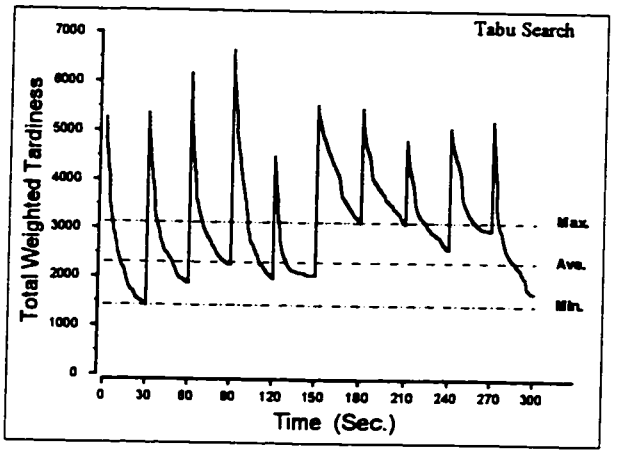
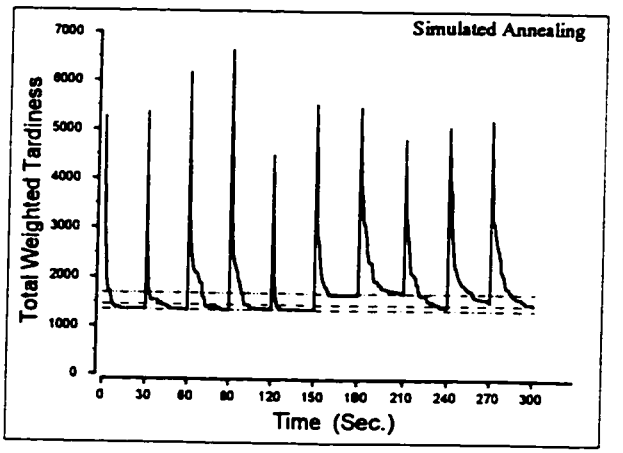


Figure 5.7(e) Comparison of convergence processes for test problem 5





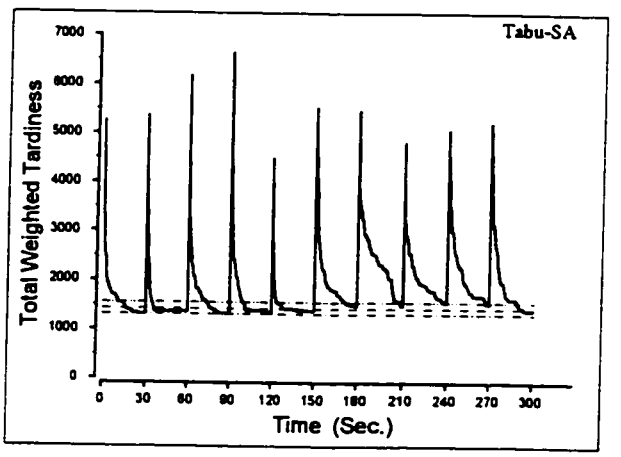
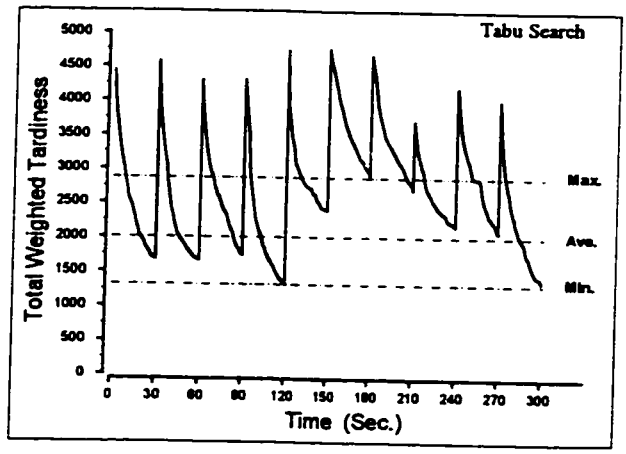
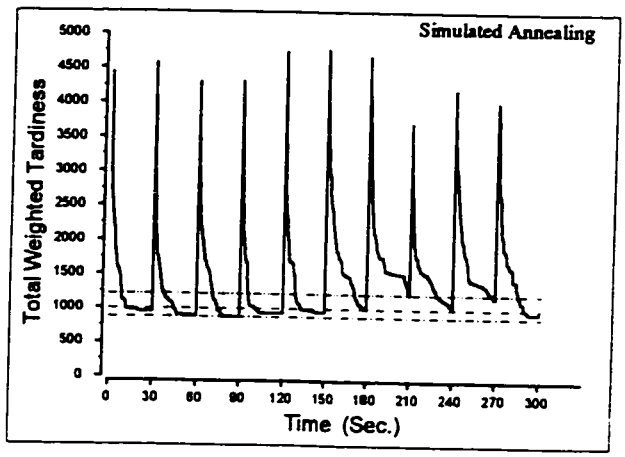


Figure 5.7(f) Comparison of convergence processes for test problem 6





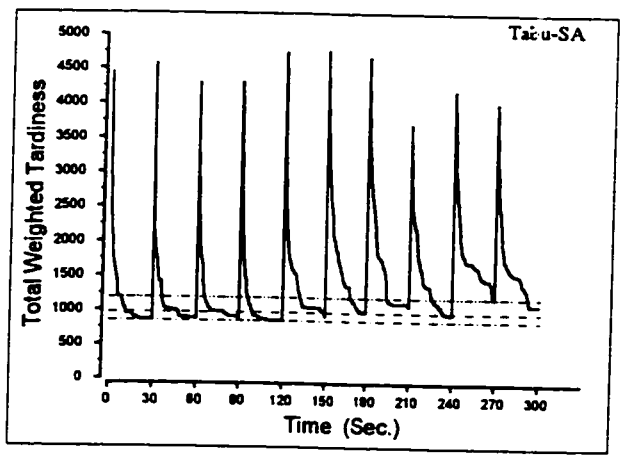
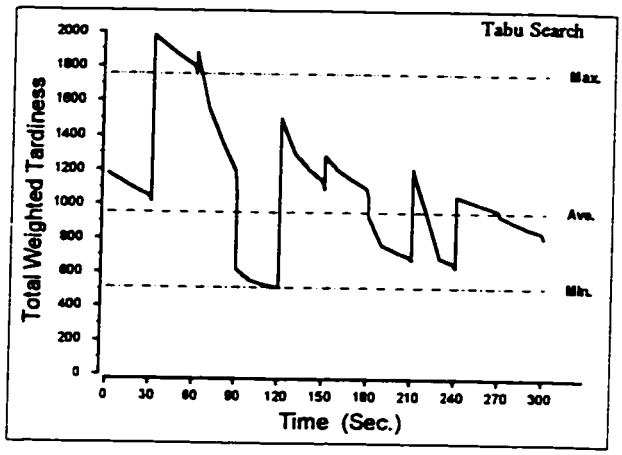
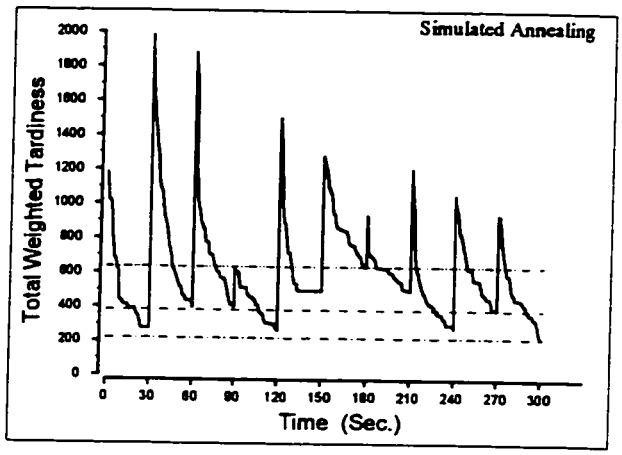


Figure 5.7(g) Comparison of convergence processes for test problem 7





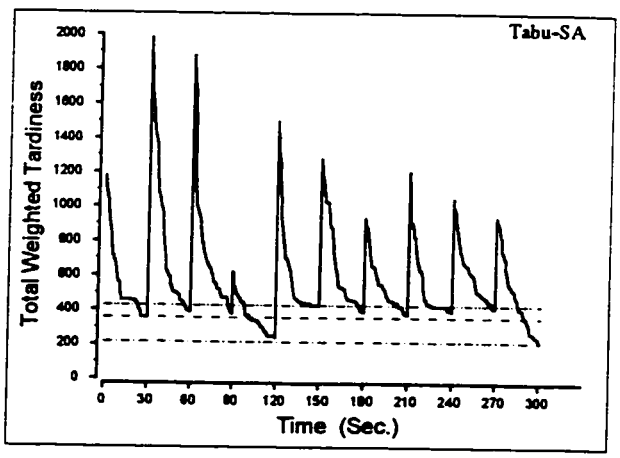
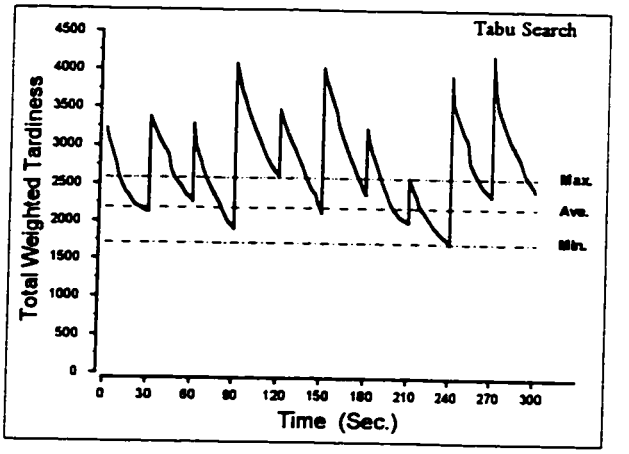
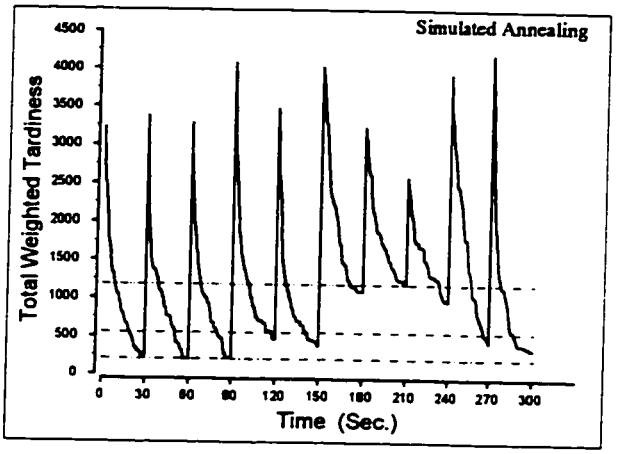


Figure 5.7(h) Comparison of convergence processes for test problem 8





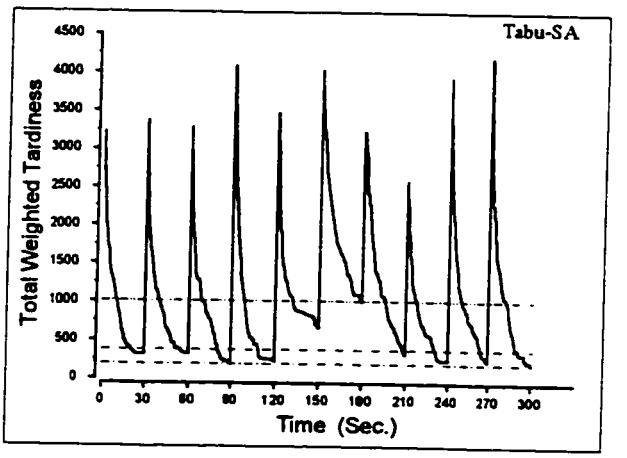
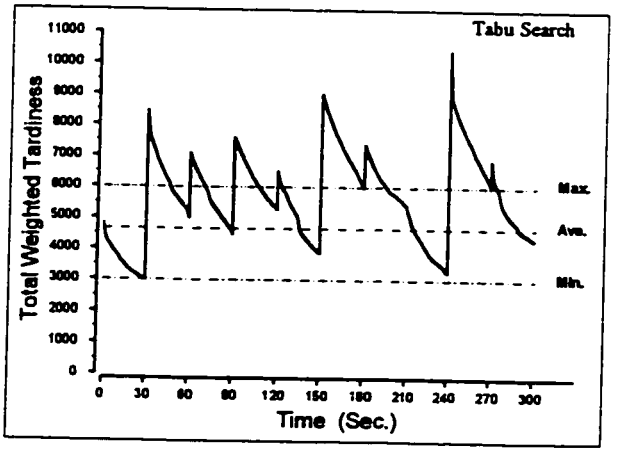
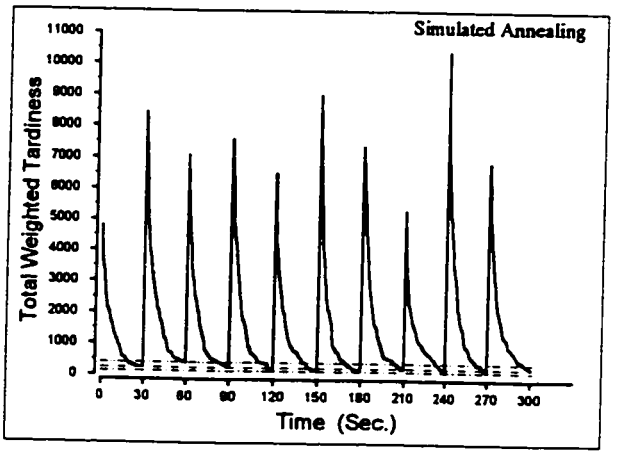


Figure 5.7(i) Comparison of convergence processes for test problem 9





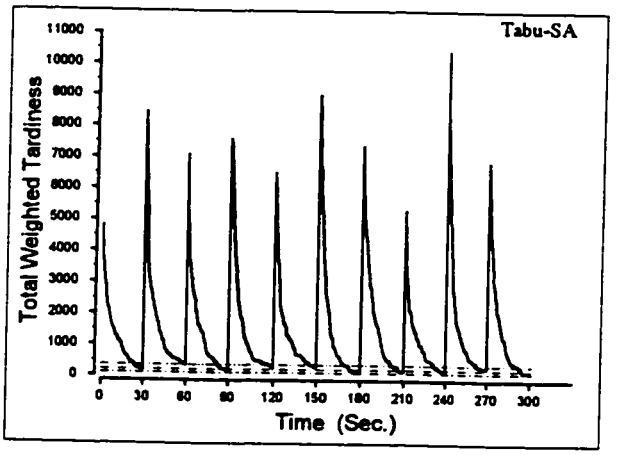


Figure 5.7(j) Comparison of convergence processes for test problem 10

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Concluding Remarks

The proposed studies on machine-part grouping and group scheduling have been carried out in Chapters 3, 4 and 5. In Chapter 3, a new structure of the Hopfield neural network, OSHN, has been designed for the binary grouping problem. The OSHN, in conjunction with an objective-guided search scheme, has been implemented in an algorithm. The solution quality of the algorithm has been tested, using 28 machine grouping problems selected from the literature. The main advantages of the proposed approach are: a) it can efficiently solve large sized problems; b) it is training-free; c) it can effectively handle the bottleneck machine problem; d) it is insensitive to the sequence of the input data and e) it is able to escape local optima and avoid cycling.

Chapter 4 presents two approaches based on simulated annealing and neural networks to the machine cell/part family formation problem considering processing time, lot size, machine capacity, and machine duplication. Many grouping efficiency measures given in the literature do not accommodate processing times, and therefore cannot reflect the real within-cell utilization and workload. A generalized grouping efficiency has been proposed to incorporate processing times. In the first approach (SA), The OSHN algorithm presented in Chapter 3 has been used to generate good seed solutions and shorten the convergence time. The application of the SA algorithm has been demonstrated using an example problem and its performance has been favourably compared with the results reported in the literature. The second approach ($OSHN_g$) employs the generalized

grouping efficiency to guide the tuning process of the network parameters and hence the search process towards the global optimum. The performance of both methods have been examined using 28 test problems. The comparison of the results shows that the SA approach is preferred when the solution quality is major concern, and the $OSHN_g$ is superior when a quick and good solution is demanded.

Chapter 5 deals with planning issues in cellular manufacturing systems. A group scheduling problem in a job shop manufacturing cell with variable machining speeds and an objective of minimizing the total weighted tardiness has been addressed. The performance of three heuristics including tabu search, simulated annealing and a new hybrid tabu-simulated annealing approach in solving such a problem has been investigated. The advantages and drawbacks of the first two heuristics have been discussed. The need for combining these two heuristics to form a hybrid approach has been justified. The three heuristics have been applied to 12 group scheduling problems. In all three algorithms, the family sequence is obtained using the MOD rule and the part sequence within each family is determined in the iterative search process. The results indicate that the hybrid tabu-simulated annealing approach outperforms the other two heuristics in terms of solution quality. The solutions obtained using the tabu-SA are also more consistent when different initial solutions are used.

6.2 Contributions

The major contributions of this thesis are as follows:

1) A new structure of Hopfield neural network (OSHN) has been designed for solving machine grouping (clustering) problems.

2) A generalized grouping efficiency index has been proposed as a performance measure for comprehensive grouping where processing times and lot sizes have

to be taken into account.

3) An objective-guided search scheme has been developed and applied to both the binary and the comprehensive grouping problems.

4) A combined tabu-SA algorithm has been proposed for solving group scheduling problems.

In addition to the above contributions, the following have also been carried out:

a) The original Hopfield neural network has been applied to the binary grouping problem.

b) A simulated annealing algorithm has been developed and applied jointly with the OSHN to the comprehensive grouping problem.

c) Two membership indexes for part assignment have been proposed for the binary and the comprehensive grouping problems.

d) A group scheduling problem with controllable machining speed in a job shop manufacturing cell has been investigated.

e) A detailed discussion on the advantages and disadvantages of the tabu search and the simulated annealing has been provided.

f) A definition for partition sets has been introduced and discussed.

6.3 Future Research Directions

The following topics could be worth exploring:

1) The energy function of the neural network presented in Chapter 3 is based on the Hopfield network. This energy function requires three parameters that have to be tuned in order to obtain a preferred solution. Developing a new neural network that directly employs the grouping efficiency as the energy function could further

improve the computational efficiency.

2) The neural network approach for comprehensive grouping ($OSHN_g$) presented in Chapter 4 is basically a binary neural network that seeks a maximum non-binary objective (i.e., the generalized grouping efficiency, η_g). Developing a neural network that works with non-binary data may result in a better solution quality.

3) The machine cell/part family formation and group scheduling problems studied in this thesis are two important issues in cellular manufacturing systems. However, there are some other issues in design and planning of the cellular manufacturing systems that could be different from those of the traditional manufacturing systems. Such issues may include tool planning, inventory control and facilities planning problems.

REFERENCES

ADENSO-DÍAZ, B., 1992, Restricted neighborhood in the tabu search for the flow shop problem, *European Journal of Operational Research*, **62**, 27-37.

ANDERBERG, M. R., 1973, *Cluster Analysis for Applications* (New York: Academic Press).

ARIZONO, I., KATO, M., YAMAMOTO, A., and OHTA, H., 1995, A new stochastic neural network model and its application to grouping and tools in flexible manufacturing systems, *International Journal of Production Research*, **33**, 1535-1548.

ASKIN, R. G., CRESSWELL, S. H., GOLDBERG, J. B., and VAKHARIA, A. J., 1991, A Hamiltonian path approach to reordering part-machine matrix for cellular manufacturing, *International Journal of Production Research*, **29**, 1081-1100.

BAKER, K. R., and KANET, J. J., 1983, Job shop scheduling with modified due dates, *Journal of Operations Management*, **4**, 11-22.

BALLAKUR, A., and STEUDEL, H. J., 1987, A within-cell utilization based heuristic for designing cellular manufacturing systems, *International Journal of Production Research*, **25**, 639-665.

BEN-ARIEH, D., and TRIANTAPHYLLOU, E., 1992, Quantifying data for group technology with weighted fuzzy features, *International Journal of Production Research*, **30**, 1285-1299.

BOCTOR, F. F., 1996, The minimum-cost, machine-part cell formation problem, *International Journal of Production Research*, **34**, 1045-1063.

BOE, W. J., and CHENG, C. H., 1991, A close neighbour algorithm for designing cellular manufacturing systems, *International Journal of Production Research*, **29**, 2097-2116.

BURBIDGE, J. L., 1963, Production flow analysis, *The Production Engineer*, **42**, 742.

BURBIDGE, J. L., 1975, *The Introduction of Group Technology* (London: Heinemann).

CARPENTER, G. A., and GROSSBERG, S., 1986, Neural dynamics of category learning and recognition: attention, memory consolidation, and amnesia. In *Brain Structure, Learning, and Memory*. R.N. Davis and E.Wegman (eds) (AAAS Symposium Series).

CARPENTER, G. A., and GROSSBERG, S., 1988, The ART of adaptive pattern recognition by a self-organizing neural network, *Computer*, **21**, 77-88.

CARRIE, A. S., 1973, Numerical taxonomy applied to group technology and plant layout, *International Journal of Production Research*, **11**, 399-416.

CHAN, H. M., and MILNER, D. A., 1982, Direct clustering algorithm for group formation in cellular manufacture, *Journal of Manufacturing Systems*, **1**, 65-75.

CHAN, D. Y., and BEDWORTH, D. D., 1990, Design of a scheduling system for flexible manufacturing cells, *International Journal of Production Research*, **28**, 2037-2049.

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.

CHANDRASEKHARAN, M. P., and RAJAGOPALAN, R., 1987, ZODIAK - An algorithm for concurrent formation of part-families and machine-cells, *International Journal of Production Research*, **25**, 835-850.

CHANDRASEKHARAN, M. P., and RAJAGOPALAN, R., 1989, Groupability: an analysis of the properties of binary data matrices for group technology, *International Journal of Production Research*, **27**, 1035-1052.

CHEN, C. Y., and IRANI, S. A., 1993, Cluster first-sequence last heuristics for generating block diagonal forms for a machine-part matrix, *International Journal of Production Research*, **31**, 2623-2647.

CHEN, C. L., COTRUVO, N. A., and BAEK, W., 1995, A simulated annealing solution to the cell formation problem, *International Journal of Production Research*, **33**, 2601-2614.

CHEN, S. J., and CHENG, C. S., 1995, A neural network-based cell formation algorithm in cellular manufacturing, *International Journal of Production Research*, **33**, 293-318.

CHENG, C. H., 1992, Algorithms for grouping machine groups in group technology. *OMEGA International Journal of Management Science*, **20**, 493-501.

CHENG, C. B., and WU, C. H., 1996, Solving the FMS part-tool grouping problem using Lagrangian relaxation approach, *the 5th Industrial Engineering Research Conference Proceedings*, 141-146.

CHOW, W. S., and HAWALESHKA, O., 1992, An efficient algorithm for solving the machine chaining problem in cellular manufacturing, *Computers and Industrial Engineering*, **22**, 95-100.

CO, H. C., and ARAAR, A., 1988, Configuring cellular manufacturing systems, *International Journal of Production Research*, **26**, 1511-1522.

DAGLI, C., and HUGGAHALLI, R., 1991, Neural network approach to group technology. In *Knowledge-Based Systems and Neural Networks: Techniques and Applications*, R. Sharda, J. Chenung, and W. Cochran (eds) (New York: Elsevier), 213-228.

DAGLI, C., and HUGGAHALLI, R., 1995, Machine-part family formation with the adaptive resonance theory paradigm, *International Journal of Production Research*, **33**, 893-913.

DAYHOFF, J. E., 1990, *Neural Network Architectures: An Introduction* (New York: Van Nostrand Reinhold).

DE WITTE, J., 1980, The use of similarity coefficients in production flow analysis, *International Journal of Production Research*, **18**, 503-514.

DUDA, R. O., and HART P. E., 1973, *Pattern Classification and Scene Analysis* (Chichester: John Wiley & Sons).

ELMARAGHY, H. A., and GU, P., 1988, Knowledge-based system for assignment of parts to machine cells, *The International Journal of Advanced Manufacturing Technology*, **3**, 33-44.

FLYNN, B. B., 1987, Repetitive lots: the use of a sequence-dependent set-up time

scheduling procedure in group technology and traditional shops, *Journal of Operations Management*, 7, 203-216.

FOO, F. C., and WAGER, J. G., 1983, Set-up times in cyclic and acyclic group technology scheduling systems, *International Journal of Production Research*, **21**, 63-73.

FRAZIER, G. V., 1996, An evaluation of group scheduling heuristics in a flow-line manufacturing cell, *International Journal of Production Research*, **34**, 959-976.

GINDY, N. N. Z., RATCHEV, T. M., and CASE, K., 1995, Component grouping for GT applications - a fuzzy clustering approach with validity measure, *International Journal of Production Research*, **33**, 2493-2509.

GLOVER, F., 1977, Heuristics for integer programming using surrogate constraints, *Decision Sciences*, **8**, 156-166.

GU, P., 1991, Application of pattern recognition and optimization to group technology, *Design, Analysis, and Control of Manufacturing Cells, ASME*, **53**, 115-128.

GUNASHINGH, K., and LASHKARI, R., 1989, Machine grouping problem in cellular manufacturing systems -- an integer programming approach, *International Journal of Production Research*, **27**, 1465-1473.

HAM, I., HITOMI, K., NAKAMURA, N., and YOSHIDA, T., 1979, Optimal group scheduling and machining-speed decision under due-date constraints, *Transactions of the ASME*, **101**, 128-134.

HAN, M.-H., and MCGINNIS, L. F., 1988, Throughput rate maximization in flexible

manufacturing cells, *IIE Transactions*, **20**, 409-417.

HARHALAKIS, G., IOANNOU, G., MINIS, I., and NAGI, R., 1994, Manufacturing cell formation under random product demand, *International Journal of Production Research*, **32**, 47-64.

HARHALAKIS, G., NAGI, R., and PROTH, J. M., 1990, An efficient heuristic in manufacturing cell formation for group technology applications, *International Journal of Production Research*, **28**, 185-198.

HITOMI, K., 1979, *Manufacturing Systems Engineering*, Chapter 4 (London: Taylor & Francis Ltd.).

HITOMI, K., and HAM, I., 1977, Group scheduling technique for multiproduct, multistage manufacturing systems, *Journal of Engineering for Industry*, 759-765.

HOPFIELD, J. J., 1982, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences USA*, **52**, 2554-2558.

HOPFIELD, J. J., and TANK, D. W., 1985, Neural computation of decisions in optimization problems, *Biological Cybernetics*, **52**, 141-152.

KAMAL, S., and BURKE, L. I., 1996, FACT: A new neural network-based clustering algorithm for group technology, *International Journal of Production Research*, **34**, 919-946.

KAO, Y., and MOON, Y. B., 1991, A unified group technology implementation using the

back propagation learning rule of neural networks, *Computers and Industrial Engineering*, **20**, 425-437.

KAPARTHI, S., and SURESH, N. C., 1991, A neural network system for shape-based classification and coding of rotational parts, *International Journal of Production Research*, **29**, 1771-1784.

KAPARTHI, S., and SURESH, N. C., 1992, Machine-component cell formation in group technology: a neural network approach, *International Journal of Production Research*, **30**, 1353-1367.

KERN, G. M., and WEI, J. C., 1991, The cost of eliminating exceptional elements in group technology cell formation, *International Journal of Production Research*, **29**, 1535-1547.

KING, J. R., 1980, Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm, *International Journal of Production Research*, **18**, 213-232.

KING, J. R., and NAKORNCHAI, V., 1982, Machine-component group formation in group technology: review and extension, *International Journal of Production Research*, **20**, 117-133.

KIRKPATRICK, S., GELATT, C. D., and VECCHI, M. P., 1983, Optimization by simulated annealing, *Science*, **220**, 671-680.

KUMAR, K. R., and VANNELLI, A., 1987, Strategic subcontracting for efficient disaggregated manufacturing, *International Journal of Production Research*, **25**, 1715-

1728.

KUMAR, C. S., and CHANDRASEKHARAN, M. P., 1990, Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology, *International Journal of Production Research*, **28**, 233-243.

KUO, H. W., and INMAN, R. R., 1990, A practical heuristic for the group technology economic lot scheduling problem, *International Journal of Production Research*, **28**, 709-722.

KUSIAK, A., 1985, The part families problem in flexible manufacturing systems, *Annals of Operations Research*, **3**, 279-300.

KUSIAK, A., 1987, The generalized group technology concept, *International Journal of Production Research*, **25**, 561-569.

KUSIAK, A., 1988, EXGT-S: A knowledge based system for group technology, *International Journal of Production Research*, **26**, 887-904.

KUSIAK, A., 1991, Branching algorithms for solving the group technology problem, *Journal of Manufacturing Systems*, **10**, 332-343.

KUSIAK, A. (ed.), 1992, *Intelligent Design and Manufacturing* (New York: Wiley).

KUSIAK, A., BOE, W. J., and CHENG, C., 1993, Designing cellular manufacturing systems: branch-and-bound and A* approaches, *IIE Transactions*, **25**, 46-56.

KUSIAK, A., and CHO, M., 1992, Similarity coefficient algorithms for solving the group

technology problem, *International Journal of Production Research*, **30**, 2633-2646.

KUSIAK, A., and CHOW, W. S., 1987a, An efficient cluster identification algorithm, *IEEE Transactions on Systems, Man, and Cybernetics*, **17**, 696-699.

KUSIAK, A., and CHOW, W. S., 1987b, Efficient solving of the group technology problem, *Journal of Manufacturing Systems*, **6**, 117-124.

KUSIAK, A., and CHUNG, Y., 1991, GT/ART: Using neural networks to form machine cells, *Manufacturing Review*, **4**, 293-301.

KUSIAK, A., and HERAGU, S. S., 1987, Group technology, *Computers in Industry*, **9**, 83-91.

KUSIAK, A., VANNELLI, A., and KUMAR, K. R., 1986, Clustering analysis: models and algorithms, *Control and Cybernetics*, **15**, 139-153.

LEE, H., and GARCIA-DIAZ, A., 1993, A network flow approach to solve clustering problems in group technology, *International Journal of Production Research*, **31**, 603-612.

LEE, S., ZHANG, C., and WANG, H. P., 1991, Fuzzy set-based procedures for machine cell formation, *Design, Analysis, and Control of Manufacturing Cells, ASME*, **53**, 31-45.

LIAO, T. W., and CHEN, L. J., 1993, An evaluation of ART1 neural models for GT part family and machine cell forming, *Journal of Manufacturing Systems*, **12**, 282-290.

LIU, C. M., and WU, J. K., 1993, Machine cell formation: using the simulated annealing

algorithm, *Journal of Computer Integrated Manufacturing*, **6**, 335-349.

LOGENDRAN, R., 1991, Effect of the identification of key machines in the cell formation problem of cellular manufacturing systems, *Computers and Industrial Engineering*, **20**, 439-449.

LOGENDRAN, R., 1992, A model duplicating bottleneck machines in the presence of budgetary limitations in cellular manufacturing, *International Journal of Production Research*, **30**, 683-694.

LOGENDRAN, R., and RAMAKRISHNA, P., 1995, Manufacturing cell formation in the presence of lot splitting and multiple units of the same machine, *International Journal of Production Research*, **33**, 675-693.

MAHMOODI, F., DOOLEY, K. J., and STARR, P. J., 1990a, An investigation of dynamic group scheduling heuristics in a job shop manufacturing cell, *International Journal of Production Research*, **28**, 1695-1711.

MAHMOODI, F., DOOLEY, K. J., and STARR, P. J., 1990b, An evaluation of order releasing and due date assignment heuristics in a cellular manufacturing system, *Journal of Operations Management*, **9**, 548-573.

MAHMOODI, F., and DOOLEY, K. J., 1991, A comparison of exhaustive and non-exhaustive group scheduling heuristics in a manufacturing cell, *International Journal of Production Research*, **29**, 1923-1939.

MAHMOODI, F., TIERNEY, E. J., and MOSIER, C. T., 1992, Dynamic group scheduling heuristics in a flow-through cell environment, *Decision Sciences*, **23**, 61-85.

MCAULEY, J., 1972, Machine grouping for efficient production, *The Production Engineer*, **51**, 53-57.

MCCORMICK, W. T., SCHWITZER, R. J., and WHITE, T. W., 1972, Problem decomposition and data reorganization by clustering techniques, *Operations Research*, **20**, 993-1009.

MOON, Y. B., and CHI, S. C., 1992, Generalized part family formation using neural network techniques, *Journal of Manufacturing Systems*, **11**, 149-160.

MOSIER, C. T., ELVERS, D. A., and KELLY, D., 1984, Analysis of group technology scheduling heuristics, *International Journal of Production Research*, **22**, 857-875.

MUKHOPADHYAY, S. K., and GOPALAKRISHNAN, A., 1995, A vector analytic (VECAN) method for solving the machine-part grouping problem in GT, *International Journal of Production Research*, **33**, 795-818.

NAGI, R., HARHALAKIS, G., and PROTH, J. M., 1990, Multiple routeings and capacity considerations in group technology applications, *International Journal of Production Research*, **28**, 2243-2257.

OKOGBAA, O. G., CHEN, M. T., CHANGCHIT, C., and SHELL, R. L., 1992, Manufacturing system cell formation and evaluation using a new inter-cell flow reduction heuristic, *International Journal of Production Research*, **30**, 1101-1118.

OZDEN, M., EGBELU, P. J., and IYER, A. V., 1985, Job scheduling in a group technology environment for a single facility, *Computers and Industrial Engineering*, **9**, 67-72.

RAO, H. A., and GU, P., 1994, Expert self-organizing neural network for the design of cellular manufacturing systems, *Journal of Manufacturing Systems*, **13**, 346-358.

RAO, H. A., and GU, P., 1995, A multi-constraint neural network for the pragmatic design of cellular manufacturing systems, *International Journal of Production Research*, **33**, 1049-1070.

RUBEN, R. A., MOSIER, C. T., and MAHMOODI, F., 1993, A comprehensive analysis of group scheduling heuristics in a job shop cell, *International Journal of Production Research*, **31**, 1343-1369.

RUSSELL, G. R., and PHILIPOOM, P. R., 1991, Sequencing rules and due date setting procedures in flow line cells with family setups, *Journal of Operations Management*, **10**, 524-545.

SEIFODDINI, H., 1989a, Duplication process in machine cells formation in group technology, *IIE Transactions*, **21**, 382-388.

SEIFODDINI, H., 1989b, Single linkage versus average linkage clustering in machine cells formation applications, *Computers and Industrial Engineering*, **16**, 419-426.

SEIFODDINI, H., 1989c, A probabilistic approach to machine cell formation in group technology, *Proceedings of the International Industrial Engineering Conference and Societies' Manufacturing and Productivity Symposium Proceedings*, Toronto, Canada, 625-629.

SEIFODDINI, H., and DJASSEMI, M., 1994, Analysis of efficiency measures for block diagonal machine-component charts, *Computers and Industrial Engineering*, **27**, 237-240.

SEIFODDINI, H., and WOLFE, P. M., 1986, Application of the similarity coefficient method in group technology, *IIE Transactions*, **18**, 271-277.

SEIFODDINI, H., and WOLFE, P. M., 1987, Selection of a threshold value based on material handling cost in machine-component grouping, *IIE Transactions*, **19**, 266-270.

SHAFER, S., KERN, G., and WEI, J., 1992, A mathematical programming approach for dealing with exceptional elements in cellular manufacturing, *International Journal of Production Research*, **30**, 1029-1036.

SKORIN-KAPOV, J., and VAKHARIA, A. J., 1993, Scheduling a flow-line manufacturing cell: a tabu search approach, *International Journal of Production Research*, **31**, 1721-1734.

SOKAL, R. R., and SNEATH, P. H. A., 1968, *Principles of Numerical Toxonomy* (New York: Freeman).

SOLOMON, M. M., MILLEN, R. A., and AFENTAKIS, P., 1995, The formation of subfamilies for machine loading of flow-line cells, *International Journal of Production Research*, **33**, 2357-2374.

SRINIVASAN, G., NARENDRAN, T. T., and MAHADEVAN, B., 1990, An assignment model for the part-families problem in group technology, *International Journal of Production Research*, **28**, 145-152.

STANFEL, L. E., 1985, Machine clustering for economic production, *Engineering Costs and Production Economics*, **9**, 73-81.

SULE, D. R., 1991, Machine capacity planning in group technology, *International Journal of Production Research*, **29**, 1909-1922.

SURESH, N. C., and KAPARTHI, S., 1994, Performance of fuzzy ART neural network for group technology cell formation, *International Journal of Production Research*, **32**, 1693-1713.

SURESH, N. C., SLOMP, J., and KAPARTHI, S., 1995, The capacitated cell formation problem: a new hierarchical methodology, *International Journal of Production Research*, **33**, 1761-1784.

VAITHIANATHAN, R., and MCROBERTS, K. L., 1982, On scheduling in a GT environment, *Journal of Manufacturing Systems*, **1**, 149-155.

VAKHARIA, A. J., and CHANG, Y. L., 1990, A simulated annealing approach to scheduling a manufacturing cell, *Naval Research Logistics*, **37**, 559-577.

VANNELLI, A., and KUMAR, K. R., 1986, A method for finding minimal bottle-neck cells for grouping part-machine families, *International Journal of Production Research*, **24**, 387-409.

VENTURA, J. A., CHEN, F. F., and WU, C. H., 1990, Grouping parts and tools in flexible manufacturing systems production planning, *International Journal of Production Research*, **28**, 1039-1056.

VENUGOPAL, V., and NARENDRAN, T., 1992, A genetic algorithm approach to the machine-component grouping problem with multiple objectives, *Computers and Industrial Engineering*, **22**, 469-480.

VENUGOPAL, V., and NARENDRAN, T., 1994, Machine-cell formation through neural network models, *International Journal of Production Research*, **32**, 2105-2116.

VISWANATHAN, S., 1995, Configuring cellular manufacturing systems: a quadratic integer programming formulation and a simple interchange heuristic, *International Journal of Production Research*, **33**, 361-376.

WAGHODEKAR, P. H., and SAHU, S., 1984, Machine-component cell formation in group technology: MACE, *International Journal of Production Research*, **22**, 937-948.

WEI, J. C., and GAITHER, N., 1990, A capacity constrained multiobjective cell formation method, *Journal of Manufacturing Systems*, **9**, 222-232.

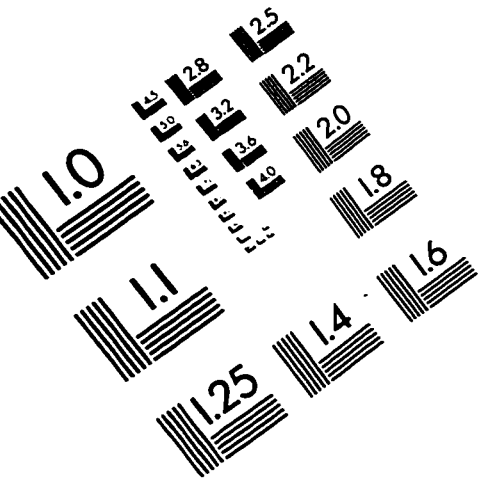
WEMMERLÖV, U., 1992, Fundamental insights into part family scheduling: the single machine case, *Decision Sciences*, **23**, 565-595.

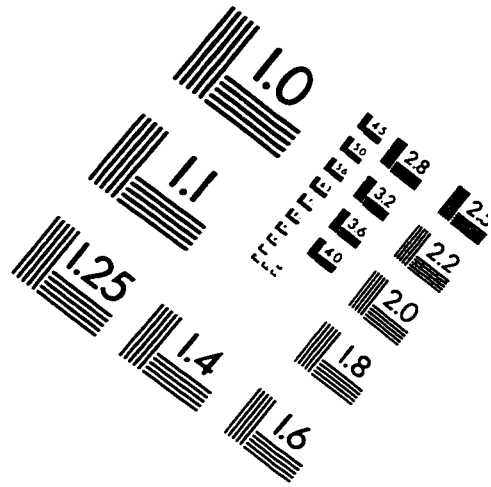
WIRTH, G. T., MAHMOODI, F., and MOSIER, C. T., 1993, An investigation of scheduling policies in a dual-constrained manufacturing cell, *Decision Sciences*, **24**, 761-788.

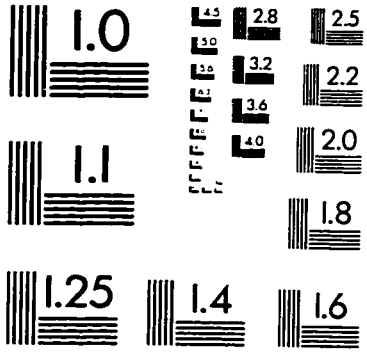
WU, N., and SALVENDY, G., 1993, A modified network approach for the design of cellular manufacturing systems, *International Journal of Production Research*, **31**, 1409-1421.

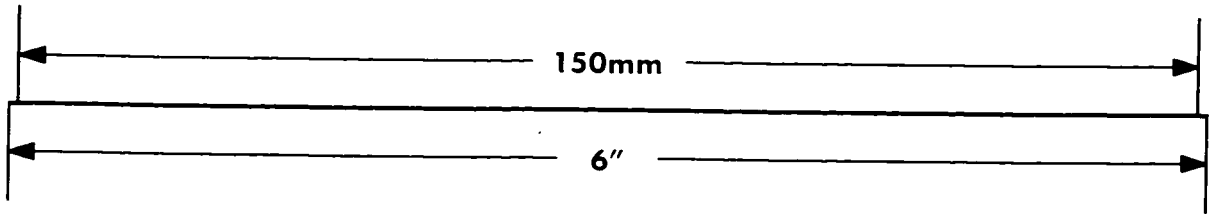
ZHANG, C., and WANG, H. P., 1992, Concurrent formation of part families and machine cells based on the fuzzy set theory, *Journal of Manufacturing Systems*, **11**, 61-67.

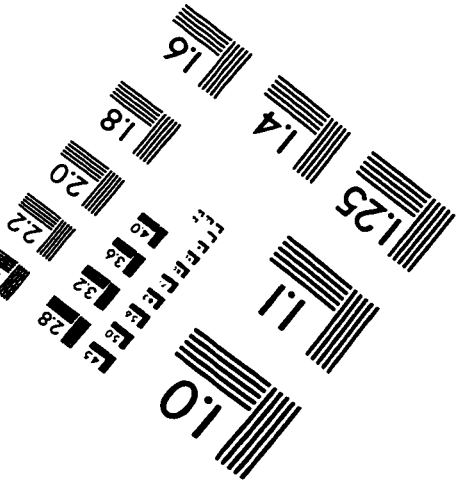
IMAGE EVALUATION TEST TARGET (QA-3)

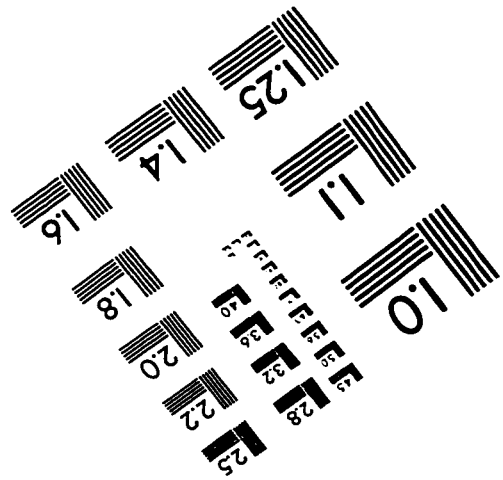












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989