AN ABSTRACT OF THE THESIS OF

Vimalin Puvanunt for the degree of Master of Science in Industrial Engineering presented on May 16, 1996. Title: The Effect of Cell Locations in the Processes of Machine Duplication and Part Subcontracting in Manufacturing Cell Design.

Redacted for privacy

Rasaratnam Logendran

Significant improvements in manufacturing productivity has resulted from implementing cellular manufacturing systems (CMSs) in small to medium sized batch-oriented manufacturing systems. The work reported in this thesis illustrates the effect of cell locations in dealing with the processes of machine duplication and part subcontracting under budgetary restrictions. In addition, this research takes into consideration the maximum number of machines assigned to a cell, which is limited by the size of cells.

The problem is formulated as a polynomial programming model and is proven to be NP-hard in the strong sense. Due to its computational complexity, a higher-level heuristic, based on a concept known as tabu-search, is proposed to efficiently solve the problem. Six different versions of the tabu search-based heuristic algorithm are tested on three different problem structures and two different layout arrangements.

The results of the research experiment concluded that the single-row layout arrangement, the tabu search-based heuristic using long term-memory based on minimal frequency (LTM_MIN) and variable tabu-list sizes is preferred to other heuristics as the

problem size increases. For the double-row layout arrangement, the tabu search based heuristic using variable tabu-list sizes and no long term-memory was found to be the efficient heuristic to search for the optimal/near-optimal solution.

The Effect of Cell Locations in the Processes of

Machine Duplication and Part Subcontracting in Manufacturing Cell Design

by

Vimalin Puvanunt

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented May 16, 1996
Commencement June 1996

Master of Science thesis of Vimalin Puvanunt presented on May 16,1996.

APPROVED:

Redacted for privacy

Major Professor, representing Industrial Engineering

Redacted for privacy

Head or Chair of Department of Industrial and Manufacturing Engineering

Redacted for privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for privacy

Vimalin Puvanunt, Author

ACKNOWLEDGMENT

Many generous people have made significant contributions to make this thesis come alive. I would like to thank them all.

My Major professor, Dr. Rasaratnam Logendran, dedicated himself to educate and guide me through the pursuit of this research. I have learned a lot in the area of cellular manufacturing systems from Dr. Logendran. He spent his invaluable time to discuss issues and provide advise to make this work possible.

I would like to thank the National Science Foundation (NSF), Washington D.C. for sponsoring this research through grant DMII-9409007. The faculty and staff of the Department of Industrial and Manufacturing Engineering to have given me the opportunity to pursue my graduate studies at OSU.

My sincere thanks go to many of my friends who took time off their busy schedules to help me out with this research. Adulwit Sonthinen helped me with the programming language. Kemmathat Vibhatavanich and Chukiat Worasucheep advised me on the UNIX system. Jakree Kuljanyawiwat assisted me with the Excustat Software. Sopin Chustasana has always given me valuable advise and encouragement. Pisut Sirikrai and Usanee Theeraveja have always been there to support me. My friends, Malinee Jimarkon and Kittisak Sukwiwat, helped me with typing the manuscript.

I would like to thank all of my committee members : Dr. Kimberly Douglas, my minor Prof., Dr. Brian Paul, my committee member, Dr. Joseph Zaworski, my Graduate Council Representative. Thanks are due the staff of the Mechanical Engineering Department for letting me use their facilities for my thesis defense.

Finally, I would like to thank those individuals for their help whose names have not been mentioned here. This thesis has been an invaluable experience in my life.

DEDICATED

TO

MY PARENTS

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES (CONTINUED)

Table Page

LIST OF TABLES (CONTINUED)

Table Page

LIST OF APPENDICES

LIST OF APPENDIX TABLES

LIST OF APPENDIX TABLES (CONTINUED)

THE EFFECT OF CELL LOCATIONS IN THE PROCESSES OF MACHINE DUPLICATION AND PART SUBCONTRACTING IN MANUFACTURING CELL DESIGN

1. INTRODUCTION

Cellular Manufacturing (CM) is regarded as an application of Group Technology which has played a significant role in improving the manufacturing productivity of small to medium sized batch-oriented manufacturing systems over the past two decades. Several benefits have been attributed to implementing CM systems including reduced setup times, reduced queue times, reduced throughput times, reduced work-in-process inventories, reduced finished good inventories, reduced labor cost, reduced material handling, better space utilization, improved production control, and simplified process planing (Jordan and Frazier 1993).

Considerable research has been reported in the published literature on the design of CM systems. The motivation for this research came about as a result of the three-phase methodology proposed by Logendran (1993). Very specifically, the research reported here is related to the third phase. Thus, prior to introducing the research problem, a brief description of each phase is provided below.

The first phase recognizes the fact that in modern manufacturing systems parts can have alternative process plans and each operation required of parts can be performed on alternative machines. Only a few studies have addressed the issue of alternative process plans in manufacturing cell design (Kusiak 1987, Choobineh 1988, Rajamani et al. 1990). Rajamani et al. developed a model which explicitly incorporated budgetary and capacity limitations into the formulation. However, the disadvantage of this model is that the budgetary limitation is set for the operating cost of producing all parts. In a deterministic manufacturing environment with stable demand patterns, it is only reasonable for the parts manufacturing company to produce the number of units demanded of each part by customers over the planning horizon. Thus, operating cost of producing the parts should not be limited by budget, rather an item that should be included in the objective function along with the amortized cost of purchasing the machines. Therefore, the first phase presented a more realistic approach to the problem of selecting a unique process plan for each part and the number of machines of each type for setting up a CM system (Logendran 1992, Logendran et al. 1994).

Once a unique process plan for each part and the number of machines of each type are determined in phase 1, the next phase focuses on identifying the assignment of parts and machines to individual manufacturing cells. There has been a significant amount of research devoted to this important problem which is referred to as the cell formation problem in the published literature (McAulley 1972, Burbidge 1977, Rajagopalan and Batra 1975, King 1980a, b, Waghodekar and Sahu 1984, Kumar et al. 1986, Kuisak 1987, Tabucanon and Ojha 1987, Ballakur and Steudel 1987, Seifoddini and Wolfe 1987, Choobineh 1988, Askin and Chia 1990, Vakharia and Wemmerlov 1990, Harhalakis et al. 1990, Rajamani et al. 1990, Chen and Inrani 1993, Kang and Wemmerlov 1993, Sankaran and Kasilingam 1993, Atmani et al. 1995, Burke and Kamal 1995, Dugli and Huggahalli 1995, Seifoddini and Djassemi 1995). The investigation in phase two (Logendran and Ramakrishna 1995) extended the previous research to include three important features: 1. Splitting the lot into two if the total workload required of a part's operation on a machine exceeded its daily unit capacity, 2. Possibility of performing two or more nonconsecutive operations of a part on the same machine, and 3. When multiple units of a machine type are considered, each unit can be assigned to a different cell if it results in reducing the material handling cost.

In most practical problems, when parts and machines are assigned to manufacturing cells, not all of the operations required of parts will be completely processed by machines assigned to the same cell. Parts that require processing by machines assigned to other cells are known as "bottleneck parts" and machines, in other cells, that process these parts are called "bottleneck machines". Published research on this subject considered either duplicating bottleneck machines or subcontracting bottleneck parts as a viable alternative to resolve these issues. Although implementing these alternatives require additional investment, in the long run, savings can be realized due to savings in material handling cost and/or machining (setup + run) time. Thus, phase 3 focused on the problem of creating disaggregated manufacturing cells, while simultaneously dealing with machine duplication and part subcontracting under budgetary restrictions (Ramakrishna 1994).

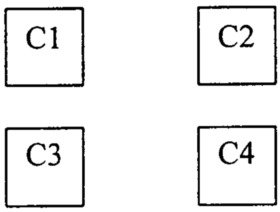
In the design of CM systems, there is clearly a need to incorporate practical design constraints (Heragu 1994). The purpose of this research is to fulfill that need by extending the phase 3 investigation to include two significant design issues: 1. The effect of cell locations in simultaneously dealing with machine duplication and part subcontracting, and 2. The maximum number of machines that can be assigned to each cell.

The effect of cell locations can further be addressed by considering the layout arrangements used for setting up manufacturing cells. Typically, two layout arrangements, linear single-row and linear double-row, are used in CM systems (Logendran 1991). Figure 1.1 shows a linear single-row layout arrangement. In this case, the most efficient movement for material-handling carriers is the movement along a straight line. If the distance between any two adjacent cells is assumed to be equal, then the distance traveled between cells 1 and 4 is three times as much as the distance traveled between cells 1 and 2. Figure 1.2 presents an alternative arrangement for the layout of cells in manufacturing systems, namely the linear double-row cellular layout. Again, if the distance between any two adjacent cells is assumed to be equal, then the distance traveled between cells 1 and 4 is the same as that traveled between any two adjacent cells. This is not the same as the distance traveled between cells 1 and 4 in a linear single-row layout. In this research, rectilinear distances are assumed for the movement by part orders between any two cells. It is, therefore, clear that not only the layout arrangement used, but also the location of each cell has a significant effect on the reduction of material handling costs which can be realized by machine duplication and part subcontracting.

Another constraint is the maximum number of machines assigned to a cell, which is limited by the size of a cell. There are a number of reasons why this is an important issue. For example, an operator can attend to a limited number of machines, floor plan dimensions may dictate the size of the cell, and to improve the utilization of operators, it may be necessary to ensure that each operator is assigned a minimum workload. In practice, the management can determine such upper and lower bounds for the number of machines that can be assigned to a cell based on past experience (Askin and Chiu 1990, Heragu 1994).

Figure 1.1 Single-row layout arrangement

Figure 1.2 Double-row layout arrangement



2. LITERATURE REVIEW

Group Technology (GT) has been known as the technique to classify the parts and machines into part families and machine cells according to their commonalties. When two or more machines are grouped together to construct machine cells used to manufacture part families, the operations performed are referred to cellular manufacturing (CM) (Offodile et al 1994). The concept of CM has been successfully applied to as many manufacturing environments.

The main objective of designing a CM system is to create machine cells, identify part families, and allocate part families to machine cells to acquire the minimum intercellular movement of parts. The extension of a variety of approaches to the cell formation problem can be categorized into three groups (Rajamani et al 1990; Heragu 1994).

- First identify machine cells and then assign parts to machine cells (Burbidge 1977, De Beer et al. 1976, 1978, De Witte 1980, McAuley 1972, Rajagopalan and Batra 1975, Faber and Carter 1986, Wemmerlov and Hyer 1986, Seifoddini and Wolfe 1987, Srinivasan et al.1990, Srinivasan and Narendran 1991).
- First identify part families and then assign machines to part families (Carrie 1973, Han and Ham 1986, Choobineh 1988, Kini et al. 1991, Offodile 1991).
- Identify machine cells and part families simultaneously (Burbidge 1971, 1973, McCormick et al. 1972, King 1980a, b, Gongaware and Ham 1981, King and Nakornchai 1982, Chan and Milner 1982, Waghodekar and Sahu 1984, Chandrasekaran and Rajagopalan 1986a, b, 1987, Kusiak and Chow 1988, Askin et al. 1991, Kusiak and Chung 1991).

Even though there is extensive research dedicated to the cell formation problem, realistically there are the "bottleneck" parts and machines which occur in relevant occasion among the formed cell. The existence of these "bottleneck" parts and machines initiate the material handling cost in terms of intercellular movement and, therefore, minimizing or eliminating "bottleneck" machines or parts should be the meaningful issue.

Seifoddini (1989) proposed the duplication procedure to reduce the number of intercellular moves in order to make the machine cells more efficient. He presents a cost-based duplication procedure which uses the duplication cost vs. an associated reduction in intercellular material handling cost to justify the decision about the duplication of machines based on economics factors. Yet, there are two limitations in this study: (i) the sequence of operations of parts, and (ii) budgetary limitations, an important managerial issue of most parts manufacturing companies, were not considered. Logendran (1990) presented a realistic two-phase methodology to describe the duplication process which overcome these limitations.

Skinner (1974) refers to the concept of focused-factories, where subcontracting has been recognized as concentrating on doing a few operations well within a plant and acquiring the rest from the outside. Thus, subcontracting bottleneck parts is the other alternative to reduce the material handling cost contributed by intercellular moves. Vannelli and Kumar (1987) have developed two efficient algorithms to identify the minimal number or minimal total cost of subcontractible parts while achieving disaggregation. This method also has the flexibility in terms of number of cells and cell size to let the designer generate a variety of cellular manufacturing systems' designs to choose from.

In addition, previous research has focused on dealing with the issues of duplicating bottleneck machines and subcontracting bottleneck parts during the formation of manufacturing cells. Kumar and Vannelli (1985) proposed a method to identify the minimum bottleneck parts and machines through either duplication of machines or subcontracting of parts while the cells are formed. Their analysis used polynomially bounded algorithms oriented toward finding minimal cut-nodes in either partition of the bipartite part-machine graph.

The issues of duplicating bottleneck machines and subcontracting bottleneck parts do not have to be dealt with, during cell formation. The paper presented by Kern and Wei (1991) documented a method for identifying opportunities for reducing the number of intercell transfers after an initial cell formation. First, their methodology recognized how each "bottleneck" part/machine in the system contributed to the creation of intercell material transfers. Subsequently, the costs associated which each alternative was analyzed to remove each bottleneck part/machine. Finally, a prioritized list of the cost for each alternative is created to suggest the most cost-effective sequence of bottleneck part/machine removal.

Also, Ramakrishna (1994) presented a model and a solution algorithm for simultaneously dealing with duplicating bottleneck machines and subcontracting bottleneck parts under budgetary restrictions. A higher-level heuristic algorithm, based on a concept known as tabu search, was implemented to efficiently solve large-size problems.

To the best of our knowledge, there has been no attempt in the past to evaluate the impact of cell locations on the method used for reducing the intercell transfers caused by the existence of bottleneck machines and parts. Moreover, the limit on the number of machines assigned to each cell has been disregarded in research previously performed in phase 3. The consideration of these issues to extend the phase 3 analysis is described in the next section under Problem Statement.

3. PROBLEM STATEMENT

Although duplicating bottleneck machines connected to each bottleneck part requires additional capital, in the long run, savings in material handling costs can be realized by not having to transport the parts between cells to complete their processing requirements. However, there is no "true" savings in machining time on bottleneck machines considered for duplication because the bottleneck parts still consume machining time on the duplicated machines in the parts' original cell assignment. The limit on the number of machines that can be assigned to each cell also has a significant impact on the extent to which bottleneck machines may be duplicated.

The other alternative to eliminate or minimize the bottleneck parts in the system is the strategy known as subcontracting the bottleneck parts to manufacturers outside the company. Subcontracting cost consists of the purchase cost of buying the same part from outside manufacturers which includes transportation, administrative and other costs associated with subcontracting. Subcontracting bottleneck parts would result not only in savings in material handling costs but also savings in machining time.

The finding from phase 3 suggests that one of three actions be taken for each bottleneck part: 1. Leave the bottleneck part as in the initial solution, 2. Duplicate all of the bottleneck machines connected to it, or 3. Subcontract the part. As pointed out before, the effect of locations and the limit on the number of machines assigned to each cell are important issues that can not be ignored when simultaneously dealing with machine duplication and part subcontracting. The cell location, in particular, has an interactive effect because the true savings in material handling cost realized due to machine duplication is dependent upon where a cell is located in relation to another. Consequently, the objectives of this research can be stated as follows.

(i) To develop a mathematical model which quantifies the effect of cell locations maximizing the total net savings obtained due to machine duplication and part subcontracting. The limit on budget and the limit on the number of machines assigned to each cell are treated explicitly as constraints in the development of the model.

(ii) To develop an efficient solution algorithm to solve the model specified in (i). The algorithm should be capable of solving industry-size problems dealing with machine duplication and part subcontracting.

In the next section, the assumptions and notations used in the development of the mathematical model as well as the model are presented. The model is formulated as a polynomial programming model. The computational complexity of the phase 3 problem was investigated and proven NP-hard in the strong sense (Ramakrishna 1994). It means that the computation time required to solve a problem would increase exponentially as the number of variables introduced in the problem increases. Thus, a fairly large problem can not be solved for its optimal solution within a reasonable computation time. The research problem considered here, being an extension of the phase 3 problem, is also NP-hard in the strong sense. Thus, a higher-level heuristic algorithm, based on a concept known as tabu search, is introduced to search for the optimum or considered nearly optimum for the problem with practical significance. Also, a simple example taken from the previous research is solved to illustrate the functionality and efficacy of the proposed tabu search-based heuristic solution algorithm.

4. MODEL DEVELOPMENT

4.1 BACKGROUND

The model developed in this research is an extension of the mathematical model proposed by Ramakrishna (1994). Ramakrishna (1994) formulated the model for simultaneously considering the effect of the role of duplicating and subcontracting processes and their interaction in reducing/eliminating "bottleneck" elements. A general and binary-integer programming model was formulated with the objective of maximizing the net savings in costs as a result of machine duplication and part subcontracting. Available machine capacities and budgetary limitation were the major constraints included in his model. However, his research did not consider the possibility of locating each cell in one of many potential locations on the shop floor. Also, the maximum number of machines assigned to each cell was not considered a constraint in his model. There is clearly a void which should be eliminated to enhance the applicability of the model for designing manufacturing cells in practice. This research investigates simultaneously the role of duplicating and subcontracting processes when alternative cell locations are present for each cell, subject to budgetary restriction. Moreover, the model considers the maximum number of machines assigned to each cell as a significant constraint in the system.

The model assumptions are presented in the next section. This is followed by the notations used in the development of the model and the description of the model parameters and constraints. Finally, a mathematical model is presented and the computational complexity of the research problem is stated.

4.2 ASSUMPTIONS

(1) The model assumes a planning horizon of one year.

(2) There are 260 work days per year as a result of having 5 work days per week, over a period of 52 weeks.

(3) The (x,y) coordinates system is used for the location of cells.

4.3 NOTATIONS

i = 1,2,...,m machines

j = 1,2,...,n parts

l = 1,2,...,c cells

m_l = numbers of bottleneck machines ($m_l \leq m$)

n_l = Number of bottleneck parts ($n_l \leq n$)

d_{qr} = distance between cells q and r; $d_{qq} = 0$

x_{ijl} = 1 if machine type i is duplicated for part j in cell l
 0 otherwise

y_{jl} = 1 if part j in cell l is subcontracted
 0 otherwise

z_{jl} = 1 if all machines connected to part j assigned to cell l is considered for duplication
 0 if part j assigned to cell l is either subcontracted or left as it was in the original solution

r_{il} = number of machines (units) of type i required to be duplicated in cell l due to capacity requirements (a general integer variable, i.e., 0,1, 2,...)

d_j = volume of production for part j measured in units per day. On a yearly basis, the number of units, $D_j = d_j * 260$.

L_j = size of unit handling load for part j measured in units

n_j = number of unit loads of part j handled per day; n = $[d_j/L_j]$ +1. On a yearly basis, the number of units loads, $N_j = n_j * 260$.

t_j = cost($) incurred in moving a unit load of part j by a unit distance

k_j = number of operations performed on part j

m(j,k) = machine (type) on which part j's kth operation is performed

c(j,k) = cell (number) in which part j's kth operation is performed

p(j,k) = sum of the setup and run times for part j's k^{th} operation on machine m(j,k). It is assumed that the required daily volume is produced in one step. Thus, p(j,k) = setup time/d_j + run time for a unit of part j's kth operation on machine m(j,k).

R_i = average cost representative of per unit machining (setup+run) time on machine i

P_l = set of parts assigned to cell l and are connected to one or more machines in other cells

PP_i = set of parts connected to machine i

M_l = set of machines assigned to cell l and are connected to one or more parts in other cells

MM_j = set of machines connected to part j

b_j = incremental cost of subcontracting a unit of part j (i.e., cost of producing a unit of part j outside - cost of producing a unit of part j in-house)

a_i = amortized cost of duplicating bottleneck machine (type) i

B = maximum dollar amount the parts manufacturing company is willing to spend over a planning horizon of one year

C_i = available capacity per each unit of machine type i on an annual basis

e_{uvl} = 1 if cell l is located in grid (u,v), where u correspond to the row # and v corresponds to the column #
  0 otherwise

r_l = number of <u>rows</u> considered in the grid/layout for locating cells

c_l = number of <u>columns</u> considered in the grid/layout for locating cells

n_l = number of units of machines assigned to cell l after the initial part and machine assignments to cells are made

N_l = maximum number of machines that can be assigned to a cell, including the duplicated machines

4.4 MODEL DESCRIPTION

The problem is formulated as a polynomial linear programming model. The formulation shown in the following mathematical model has an objective function which focuses on maximizing the net savings in costs due to simultaneously considering machine duplication and part subcontracting, when alternative cell locations for each cell are available in the system. The factors considered in the development of the objective function deserve an explanation. The sequence of operations of the bottleneck part is a significant factor in evaluating the material handling costs incurred by intercellular moves. In practice, there is no guarantee that the first operation of the bottleneck part will be performed on a machine assigned to the same cell as that of the part. Taking these factors into consideration, the saving in material handling cost due to duplicating bottleneck machines connected to bottleneck part j assigned to cell l is evaluated as

$$\sum_{l=1}^{C} \sum_{j \in P_l} \Big[N_j \cdot t_j \cdot \Big\{ d_{lc(j,1)} \cdot x_{m(j,k)jl} \cdot \sum_{u_2=1}^{r_1} \sum_{v_1=1}^{t_1} e_{u_1 v_1 l} \cdot \sum_{u_2=1}^{r_1} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,1)} \cdot \Big\{ |v_2 - v_1| \text{ if } u_1 = u_2 + |u_2 - u_1| \text{ if } v_1 = v_2$$

$$+ |(v_2 - v_1) + (u_2 - u_1)| \text{ if } u_1 \neq u_2 \text{ and } v_1 \neq v_2 \Big\}$$

$$+ \sum_{k=1}^{k_j - 1} d_{c(j,k),c(j,k+1)} \cdot (x_{m(j,k)jl} | \text{ if } c(j,k+1) = l + x_{m(j,k+1)jl} | \text{ if } c(j,k+1) \neq l) \cdot \sum_{u_1=1}^{r_l} \sum_{v_1=1}^{t_1} e_{u_1 v_1 c(j,k)} \cdot \sum_{u_2=1}^{r_l} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,k+1)}$$

$$\cdot \Big\{ |v_2 - v_1| \text{ if } u_1 = u_2 + |u_2 - u_1| \text{ if } v_1 = v_2 + |(v_2 - v_1) + (u_2 - u_1)| \text{ if } u_1 \neq u_2 \text{ and } v_1 \neq v_2 \Big\} \Big]$$

The amortized cost of duplicating bottleneck machines connected to bottleneck part j assigned to cell l is evaluated as

$$\sum_{l=1}^{c} \sum_{j \in P_l} \left\{ \sum_{i \in MM_j} x_{ijl} \cdot a_i \right\}$$

The net saving contributed by duplicating bottleneck machines connected to a bottleneck part can be evaluated by subtracting the amortized cost from saving due to duplication identified above. Now consider a situation where two bottleneck parts are assigned to the same cell l, and connected to a common bottleneck machine. In order to evaluate the total net savings for each part, the amortized cost of the bottleneck machine would have been subtracted from its savings. However, to evaluate the total savings for both parts, the amortized cost should have been subtracted only once, provided the available manned hours on the bottleneck machine is sufficient to meet the processing time requirements of both parts. Therefore, the amortized cost of the bottleneck machine must have been added as an adjustment to the total net savings for both parts to compensate for the double counting performed in the evaluation of the total net savings for each part separately. Thus the term, called adjustment is evaluated as:

$$\sum_{l=1}^{c} \sum_{\substack{i \in M_s \\ and \\ s \neq l}} \left[\sum_{j \in P_l \cap PP_i} \left(x_{ijl} \right) - r_{il} \right] \cdot a_i$$

The saving in material handling cost due to subcontracting bottleneck part j assigned to cell l is evaluated as

$$\sum_{l=1}^{c} \sum_{j \in P_l} \Big[N_j \cdot t_j \cdot \Big\{ d_{lc(j,1)} \cdot \sum_{u_2=1}^{r_1} \sum_{v_1=1}^{t_1} e_{u_1 v_1 l} \cdot \sum_{u_2=1}^{r_1} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,1)} \cdot \Big\{ \left| v_2 - v_1 \right| \; if \; u_l = u_2 + \left| u_2 - u_1 \right| \; if \; v_1 = v_2 + \left| (v_2 - v_1) + (u_2 - u_1) \right| \; if \; u_l \neq u_2 \; and \; v_1 \neq v_2 \Big\}$$

$$+$$

$$\sum_{k=1}^{k_j - 1} d_{c(j,k),c(j,k+1)} \cdot \sum_{u_1=1}^{r_l} \sum_{v_1=1}^{t_1} e_{u_1 v_1 c(j,k)} \cdot \sum_{u_2=1}^{r_l} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,k+1)} \cdot \Big\{ \left| v_2 - v_1 \right| \; if \; u_l = u_2 + \left| u_2 - u_1 \right| \; if \; v_1 = v_2$$

$$+ \left| (v_2 - v_1) + (u_2 - u_1) \right| \; if \; u_l \neq u_2 \; and \; v_1 \neq v_2 \Big\} \Big\}$$

The saving due to subcontracting bottleneck part j (assigned to cell l) contributed by machining time saved on all of the machines where the part's operations are scheduled to be performed is evaluated as

$$\sum_{l=1}^{c} \sum_{j \in P_l} \left[D_j \cdot \sum_{k=1}^{k_j} p_{(j,k)} \cdot R_{m(j,k)} y_{jl} \right]$$

The incremental cost associated with subcontracting bottleneck part j assigned to cell l is evaluated as

$$\sum_{l=1}^{c} \sum_{j \in P_l} D_j b_j y_{jl}$$

Therefore the objective function, focusing on maximizing the total net saving in costs for all bottleneck parts is presented in the next section. The constraints considered in the model are described below, and are presented under the objective function in the next section.

Constraints (1) and (2) ensure that each cell can only occupy one location.
Constraint (3) ensures that the total number of machines assigned to each cell does not exceed the maximum limitation.
Constraint (4) ensures that the total amount spent on duplicating bottleneck machines and subcontracting bottleneck parts be within the budgetary limitation (B) specified by the parts manufacturing company.
Constraint (5) ensures a feasible capacity is maintained on those bottleneck machines chosen for duplication, assuming that the bottleneck machines chosen are currently not included in the bottleneck part's home cell.
Constraints (6), (7), (8) and (9) specifically impose the requirement that a bottleneck part is left to remain a bottleneck as it was in the initial solution, subcontracted, or all of the bottleneck machines connected to it are duplicated.

4.5 MATHEMATICAL MODEL

Maximize

$$Z = \sum_{l=1}^{c} \sum_{j \in P_l} \Big[N_j \cdot t_j \cdot \Big\{ d_{lc(j,1)} \cdot x_{m(j,k)jl} \cdot \sum_{u_2=1}^{r_1} \sum_{v_1=1}^{t_1} e_{u_1 v_1 l} \cdot \sum_{u_2=1}^{r_1} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,1)} \cdot \Big\{ \left| v_2 - v_1 \right| \; if\; u_l = u_2 + \left| u_2 - u_1 \right| \; if\; v_1 = v_2$$

$$+ \left| (v_2 - v_1) + (u_2 - u_1) \right| \; if\; u_l \neq u_2 \; and\; v_1 \neq v_2 \Big\}$$

$$+ \sum_{k=1}^{k_j - 1} d_{c(j,k),c(j,k+1)} \cdot \left(x_{m(j,k)jl} \right| if\; c(j,k+1) = l + x_{m(j,k+1)jl} \left| \; if\; c(j,k+1) \neq l \right) \cdot \sum_{u_1=1}^{r_l} \sum_{v_1=1}^{t_1} e_{u_1 v_1 c(j,k)} \cdot \sum_{u_2=1}^{r_l} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,k+1)}$$

$$\cdot \Big\{ \left| v_2 - v_1 \right| \; if\; u_l = u_2 + \left| u_2 - u_l \right| \; if\; v_l = v_2 + \left| (v_2 - v_l) + (u_2 - u_l) \right| \; if\; u_l \neq u_2 \; and\; v_l \neq v_2 \Big\} \Big\} - \sum_{i \in MM_j} x_{ijl} \cdot a_i \Big]$$

$$+ \sum_{l=1}^{c} \sum_{\substack{i \in M_s \\ and \\ s \neq l}} \left[\sum_{j \in P_l \cap PP_i} (x_{ijl}) - r_{il} \right] \cdot a_i$$

$$+ \sum_{l=1}^{c} \sum_{j \in P_l} \Big[N_j \cdot t_j \cdot \Big\{ d_{lc(j,1)} \cdot \sum_{u_2=1}^{r_1} \sum_{v_1=1}^{t_1} e_{u_1 v_1 l} \cdot \sum_{u_2=1}^{r_1} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,1)} \cdot \Big\{ \left| v_2 - v_1 \right| \; if\; u_l = u_2 + \left| u_2 - u_1 \right| \; if\; v_1 = v_2$$

$$+ \left| (v_2 - v_1) + (u_2 - u_1) \right| \; if\; u_l \neq u_2 \; and\; v_1 \neq v_2 \Big\}$$

$$+ \sum_{k=1}^{k_j - 1} d_{c(j,k),c(j,k+1)} \cdot \sum_{u_1=1}^{r_l} \sum_{v_1=1}^{t_1} e_{u_1 v_1 c(j,k)} \cdot \sum_{u_2=1}^{r_l} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,k+1)} \cdot$$

$$\Big\{ \left| v_2 - v_1 \right| \; if\; u_l = u_2 + \left| u_2 - u_1 \right| \; if\; v_1 = v_2 + \left| (v_2 - v_1) + (u_2 - u_1) \right| \; if\; u_l \neq u_2 \; and\; v_1 \neq v_2 \Big\}$$

$$+ D_j \cdot \sum_{k=1}^{k_j} p(j,k) \cdot R_{m(j,k)} - D_j b_j \Big] y_{jl}$$

subject to:

$$\sum_{u=1}^{r_1} \sum_{v=1}^{t_1} e_{uvl} = 1 \qquad ;\; l = 1, \ldots, c \qquad (1)$$

$$\sum_{l=1}^{c} e_{uvl} = 1 \qquad ;\; u = 1, \ldots, r_1 \qquad (2)$$

$$v = 1, \ldots, t_1$$

$$n_l + \sum_{j \in P_l} \sum_{i \in MM_j} x_{ijl} - \sum_{\substack{i \in M_s \\ s \neq l}} \left[\sum_{j \in P_l \cap PP_i} (x_{ijl}) - r_{il} \right] \leq N_l \tag{3}$$

$$\sum_{l=1}^{c} \sum_{j \in P_l} \sum_{i \in MM_j} a_i \cdot x_{ijl} - \sum_{l=1}^{c} \sum_{\substack{i \in M_s \\ s \neq l}} \left[\sum_{j \in P_l \cap PP_i} (x_{ijl}) - r_{il} \right] a_i + \sum_{l=1}^{c} \sum_{j \in P_l} D_j b_j y_{il} \leq B \tag{4}$$

$$260 \cdot \sum_{\substack{i \in M_s \\ and \\ s \neq l}} \sum_{j \in P_l \cap PP_i} \cdot \sum_{\substack{k=1,\ldots,k_j \\ c(j,k) \neq l}} p(j,k) \cdot x_{m(j,k)jl} < C_i r_{il} \qquad ; l = 1,2,\ldots,c \tag{5}$$

<u>either</u> $$y_{il} \leq 1 \tag{6}$$

<u>or</u> $$\sum_{i \in MM_j} x_{ijl} = |MM_j| \quad \underline{\text{or}} \quad 0 \qquad \begin{matrix} l = 1,2,\ldots,c \\ j \in P_l \end{matrix} \tag{7}$$

$$y_{jl} + z_{jl} \leq 1 \tag{8}$$

$$\sum_{i \in MM_j} x_{ijl} - |MM_j| \cdot z_{jl} = 0 \tag{9}$$

4.6 COMPUTATIONAL COMPLEXITY OF THE RESEARCH PROBLEM

The mathematical model developed above is a polynomial programming model. The model presented in Ramakrishna(1994) can be considered a special case of the model proposed above because his research did not take into consideration the effect of alternative cell locations. The complexity of that problem was investigated and proven NP-hard in the strong sense (Logendran and Ramakrishna 1993). As the special case is already strongly NP-hard, it can be concluded that the research problem investigated here is also NP-hard in the strong sense.

Even for the special case (Ramakrishna 1994) the possibility of employing an implicit search algorithm such as the branch-and-bound technique is ruled out as such algorithms would turn out to be too time consuming even for a problem with moderate number of bottleneck machines and bottleneck parts. Therefore, a higher-level heuristic algorithm based on a concept known as tabu search was introduced to solve large-scale problems encountered in industry practice. Following this lead, a tabu search-based heuristic algorithm has been proposed in the next chapter to efficiently solve large problems.

5. HEURISTIC ALGORITHM

5.1 INTRODUCTION

Tabu search-based algorithms have been successfully implemented to obtain optimal or near optimal solutions to a wide variety of combinatorial problems including employee scheduling (Glover and Mc Millan 1986), space planing and architectural design (Glover et al. 1985), job shop scheduling (Eck 1989), machine scheduling (Laguna et al. 1989), quadratic assignment problems (Skorin-Kapov 1989), traveling salesman problems (Knox 1989, Malek et al. 1989, Heap et al. 1989), and a variety of other problems. The core of Tabu search is its capability of overcoming the problem of being trapped in a local optimum, if one was encountered, during the search. Tabu search uses flexible memory structures (to permit search information to be exploited more thoroughly than by the rigid memory structures), conditions for strategically constraining and freeing the search process (incorporated in tabu restrictions and aspiration criteria), and memory functions of varying time spans for intensifying and diversifying the search (by reinforcing good history attributes and driving the search into new regions)(Glover 1990).

The motivation for developing a tabu-search based heuristic algorithm for solving the problem addressed in this research are the computational complexity of the problem considered which is shown NP-hard in the strong sense, and the property of tabu search itself which has been proven to find the optimal or near optimal solution within a reasonable time span.

5.2 MECHANISM

The problem investigated focuses on allocating each cell to a location while simultaneously dealing with the processes of machine duplication and part subcontracting. In this context, each cell can be located in one of several locations of a given layout (alternative cell locations) and for a solution representing specific cell locations, each bottleneck part can be considered for one of three options (alternative part options) -subcontracting, duplicating machines connected to it, or neither. The tabu search based heuristic is applied to find the optimal/near-optimal solution in both levels of the problem; the alternative cell locations considered as outside search, and the alternative part options considered as inside search.

The final solution for the problem is composed of the solution corresponding to optimal/near-optimal cell locations together with the solution corresponding to optimal/near-optimal part options that contribute to the maximum total net saving for the entire system. The tabu search-based heuristic is applied to the outside search to move from a solution representing specific cell locations to another. For the inside search, it is applied to move from one part option to another. Once the outside search is performed to obtain the solution representing specific cell locations, the search process is switched to the inside to search for the optimal/near optimal part options as well as the resulting total net saving in costs for the outside solution. The search process is then switched back to the outside search to find a new and better solution representing specific cell locations. In general, the entire search performed by the outside search and the inside search is recognized as the evaluation procedure for the outside search. In other words, the inside search is a subset of the outside search which is the navigator of the entire search process. The flow chart shown in Figure 5.1 illustrates the mechanisms incorporated in the tabu search-based procedure. The pseudo code is provided in appendix E.1, as well.

5.3 STEPS ASSOCIATED WITH THE HEURISTIC ALGORITHM

Notation:

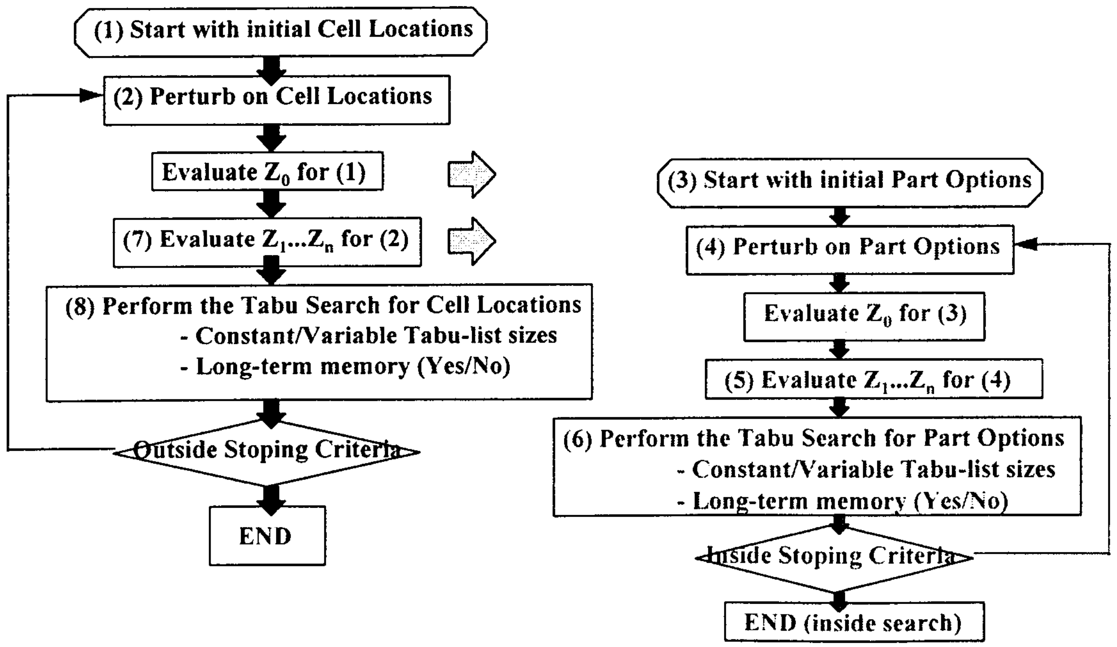
A feasible solution (FS) for the problem considered here consists of a sequence of cell locations called FSc and a sequence of bottleneck part options for the given cell location FSc called FSp. Two different sets of seeds considered for such a feasible solution are defined as follows.

- Sc(FSc) = [FS′c: FS′c is a sequence of cell locations obtained from FSc by perturbing on each location, but one location at a time. The perturbation is performed by swapping the cell occupying a location with the cell next to it, occupying a location in the nearest neighborhood.]

- Sp(FSp) = [FS′p : FS′p is a sequence of alternative part options corresponding to a given cell location FSc. FS′p is obtained from FSp by perturbing on options for each bottleneck part, yet one bottleneck part at a time. The perturbation on a bottleneck part option is to change the current bottleneck part option to an alternative part option for every bottleneck part.

The steps associated with the tabu search-based heuristic algorithm can be summarized as follows:

Figure 5.1 The mechanisms incorporated in the tabu search-based procedure.



Step 1: Generate the outside initial solution for alternative cell locations. [loc_1, loc_2 ,..., loc_N] where loc_i denotes the location for cell i and N is the total number of cells in the manufacturing system. In this problem the outside initial cell location is simply assumed as [1,2,...,N].

Step 2: Using the outside initial feasible solution (FSc_0) generated in step 1 as a node, perform the outside perturbations to completely evaluate its seeds Sc(FSc_0) by perturbing on each location, one location at a time. In other word, when the cell occupying a location is swapped with the cell next to it, occupying a location in the nearest neighborhood, the other cells remain at their original cell locations. The result of the perturbation is a sequence of locations which are considered the seeds of the initial cell locations configuration, Sc(FSc_0).

Step 3: Evaluate the optimal/near-optimal total saving for outside initial feasible solution (FSc_0) obtained from step 1 by performing an inside search. The inside search is initiated by finding the initial solution listed below. This procedure involves the 10 evaluations listed below.

1). Evaluate the total savings contributed by duplicating the bottleneck machines connected to bottleneck part j assigned to cell l as DU_{jl}, where

$$DU_{jl} = \sum_{l=1}^{c} \sum_{j \in P_l} \Big[N_j \cdot t_j \cdot \Big\{ d_{lc(j,1)} \cdot x_{m(j,k)jl} \cdot \sum_{u_2=1}^{r_1} \sum_{v_1=1}^{t_1} e_{u_1 v_1 l} \cdot \sum_{u_2=1}^{r_1} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,1)} \cdot \big\{ |v_2 - v_1| \; if \; u_1 = u_2 + |u_2 - u_1| \; if \; v_1 = v_2$$
$$+ |(v_2 - v_1) + (u_2 - u_1)| \; if \; u_1 \neq u_2 \; and \; v_1 \neq v_2 \Big\}$$
$$+ \sum_{k=1}^{k_j - 1} d_{c(j,k),c(j,k+1)} \cdot (x_{m(j,k)jl} | \; if \; c(j,k+1) = l + x_{m(j,k+1)jl} | \; if \; c(j,k+l) \neq l) \cdot \sum_{u_1=1}^{r_l} \sum_{v_1=1}^{t_1} e_{u_1 v_1 c(j,k)} \cdot \sum_{u_2=1}^{r_l} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,k+1)}$$
$$\cdot \big\{ |v_2 - v_1| \; if \; u_1 = u_2 + |u_2 - u_1| \; if \; v_1 = v_2 + |(v_2 - v_1) + (u_2 - u_1)| \; if \; u_1 \neq u_2 \; and \; v_1 \neq v_2 \; \big\} \Big]$$

2). Evaluate the cost of duplicating the bottleneck machines connected to bottleneck part j assigned to cell l as E_{jl}, where

$$E_{jl} = \sum_{i \in MM_j} x_{ijl} \cdot a_i$$

3). Evaluate the savings contributed by subcontracting bottleneck part j assigned to cell l as SC_{jl}, where

$$SC_{jl} = \sum_{l=1}^{c} \sum_{j \in P_l} \Big[N_j \cdot t_j \cdot \Big\{ d_{kc(j,1)} \cdot \sum_{u_2=1}^{r_1} \sum_{v_1=1}^{t_1} e_{u_1 v_1 l} \cdot \sum_{u_2=1}^{r_1} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,1)} \cdot \{ |v_2 - v_1| \text{ if } u_l = u_2 + |u_2 - u_l| \text{ if } v_1 = v_2$$
$$+ |(v_2 - v_1) + (u_2 - u_1)| \text{ if } u_l \neq u_2 \text{ and } v_1 \neq v_2 \}$$

$$+ \sum_{k=1}^{k_j - 1} d_{c(j,k),c(j,k+1)} \cdot \sum_{u_1=1}^{r_l} \sum_{v_1=1}^{t_1} e_{u_1 v_1 c(j,k)} \cdot \sum_{u_2=1}^{r_l} \sum_{v_2=1}^{t_1} e_{u_2 v_2 c(j,k+1)} \cdot \{ |v_2 - v_1| \text{ if } u_l = u_2 + |u_2 - u_l| \text{ if } v_1 = v_2$$
$$+ |(v_2 - v_1) + (u_2 - u_1)| \text{ if } u_l \neq u_2 \text{ and } v_1 \neq v_2 \}\Big\}$$

$$+ \sum_{l=1}^{c} \sum_{j \in P_l} \Big[D_j \cdot \sum_{k=1}^{k_j} p(j,k) \cdot R_{m(j,k)} y_{jl} \Big]$$

4). Evaluate the incremental cost associated with subcontracting part j assigned to cell l as F_{jl}, where

$$F_{jl} = \sum_{l=1}^{c} \sum_{j \in P_l} D_j b_j y_{jl}$$

5). Evaluate the net savings due to duplicating the machines connected to bottleneck part j as NDU_{jl} and the net saving due to subcontracting the part itself as NSC_{jl}, where

$NDU_{jl} = DU_{jl} - E_{jl}$; $j \in P_l$ and $l = 1, 2, \ldots, c$

$NSC_{jl} = SC_{jl} - F_{jl}$; $j \in P_l$ and $l = 1, 2, \ldots, c$.

6). Evaluate maximum contribution due to each bottleneck part as $MAXCON_{jl}$:

$MAXCON_{jl} = \text{Max}\,[NDU_{jl}, NSC_{jl}]$

If $MAXCON_{jl} < 0$, neither duplication of bottleneck machines nor subcontracting of bottleneck part will be considered for part j at the *present* time. Thus, $MAXCON_{jl}$ will be set equal to zero. Later, duplicating machines connected to part j may be found attractive when some or all of these bottleneck machines have been duplicated due to their connections to other parts assigned to the same cell. Yet, a part currently found unattractive for subcontracting would remain unattractive throughout the search.

At this point, the inside initial feasible solution (FSp_0) is evaluated by assigning the option that contributes the most $MAXCON_{jl}$ to each bottleneck part. Moreover, the total savings for the initial solution denoted by TS is evaluated as:

$$TS = \sum MAXCON_{jl} \quad ; j \in P_l \text{ and } l = 1, 2, ..., c$$

7). To evaluate the objective function value Z for the inside initial solution denoted by Z_{iint}, an adjustment term should be added to TS where:

$$\text{adjustment} = \sum_{\substack{l=1 \\ and \\ s \neq l}}^{c} \sum_{i \in Ms} \left[\sum_{j \in P_l \cap PP_i} (x_{ijl}) - r_{il} \right] \cdot a_i$$

8). To deal with the budgetary restriction, first the total expenses (E) is evaluated as :

$$E = \sum_{l=1}^{c} \sum_{j \in P_l} \{ \sum_{\substack{l=1 \\ and \\ s \neq l}}^{c} \sum_{i \in Ms} \left[\sum_{j \in P_l \cap PP_i} (x_{ijl}) - r_{il} \right] \cdot a_i \} + \sum_{l=1}^{c} \sum_{j \in P_l} D_j b_j y_{jl}$$

If (E-B) is positive, the budgetary restriction would be violated. For every dollar violated, the objective function is penalized by 10 monetary units. The 10 monetary units used as penalty is not critically important. The point is that the value chosen for penalty should be large enough to make the over budgetary solution highly unattractive compared to other feasible solutions. The corresponding penalty is evaluated as:

Penalty 1	= 10 (E-B)	if (E-B) > 0
	= 0	otherwise

9). To ensure that the limitation on the number of machines assigned to each cell is met, the number of machines currently assigned to each cell is compared with the maximum allowable number. A penalty would be incurred if the total number of machines currently assigned to a cell exceeded the maximum. In this research, a penalty of 50000 monetary units is used for every machine that exceeded the maximum limit in each cell.

Again, the monetary value assumed for the penalty should be sufficiently large to make an infeasible solution unattractive in comparison to other feasible solutions.

Penalty 2 that is assigned to penalize the infeasible solution is evaluated as:

$$\text{Penalty 2} = \sum_{l=1}^{c} 50000\,(n_l - N_l) \qquad \text{if } (n_l - N_l) > 0$$

$$= 0 \qquad \text{otherwise}$$

10). Considering the adjustment term, the budgetary limitation, and the maximum number of machines that can be assigned to each cell, the total net saving for the inside initial solution (Z_{iint}) is evaluated as:

$Z_{iint} = Z_{iint}$ + adjustment - Penalty 1 - Penalty 2

Step 4: Use the inside initial solution (FSp_0) obtained in step 3 as a node, to completely evaluate its seeds $Sp(FSp_0)$ by perturbing on a bottleneck part's options for each bottleneck part, but one part at a time. When an option of a bottleneck part is being perturbed, the other bottleneck parts remain at their current options. This perturbation is performed in the ascending order of bottleneck parts. That is, bottleneck part 1(bp_1) would be perturbed first, followed by bp_2, bp_3 and so on. The results of this perturbation is the set of different bottleneck parts' options which are considered the seeds of the inside initial solution (FSp_0).

Step 5: Evaluate the total saving (Z) for each seeds obtained in step 4. The evaluation procedure is similar to the procedure described in step 3 except the sixth evaluation which evaluates the maximum contribution due to each bottleneck part as $MAXCON_{jl}$. Instead, the net saving contributed by each bottleneck part according to the bottleneck part's option configuration (CON_{jl}) is now evaluated.

$CON_{jl} = NDU_{jl}$ when part j takes the duplicating option

$= NSC_{jl}$ when part j takes the subcontracting option

$= 0$ when part j takes neither

As a result, the total saving due to the parts' option configurations, TS, can be evaluated as :

$TS = \sum CON_{jl}$; $j \in P_l$ and $l = 1, 2, ..., c$

Step 6: Perform tabu search for parts' option (inside) level to find the optimal/near-optimal parts' option configuration by moving from the inside initial configuration (FSp_0) to the "best" candidate among its seeds. This move is called the in_move (or in_iteration). The value of move is evaluated as the total saving after the move - total saving before the move. Thus an improving move would have a positive value of move. At each iteration, the parameters considered for the inside tabu search have to be updated as follows:

(1) Inside_tabu list (in_tabu list)

The in-tabu list is a parameter because it is used as a list to prevent the search being performed by perturbing on a part's option that was most recently perturbed. Whenever an in_move is performed, the in_tabu list is updated by moving into this list the bottleneck part that is being perturbed and its original option. The bottleneck parts with their original options that appear in the in_tabu list indicate that these options have been chosen for the corresponding bottleneck parts before at some previous iterations. At the present iteration these particular bottleneck parts are not allowed to move to the history options that are still in the in_tabu list unless an aspiration criterion which allows the tabu status to be overridden is satisfied.

A list of a parts and their history options remain in the in_tabu list only a certain number of iterations determined by the in_tabu list size. The in_tabu list is updated circularly according to its size. It means that if the in_tabu list was stored up to its size, the oldest entry must be removed before the next entry is stored. Two types of tabu list sizes are considered in this research; the fixed tabu-list size and the variable tabu-list size. Based on preliminary experimentation the tabu-list size is evaluated as follows:

- The fixed tabu list size for the inside search is determined by the following formula

The fixed size of in_tabu list $= \lfloor (N*K)/5 \rfloor$, if (N*K)/5 is a real number with a decimal value < 0.5

$= \lceil (N*K)/5 \rceil$, if (N*K)/5 is a real number with a decimal value ≥ 0.5

- The variable tabu-list sizes for the inside search are determined by the following formulae.

The initial size of in_tabu list $= \lfloor (N*K)/5 \rfloor$, if (N*K)/5 is a real number with a decimal value < 0.5

$= \lceil (N*K)/5 \rceil$, if (N*K)/5 is a real number with a decimal value ≥ 0.5

The decreased size of in_tabu list $= \lfloor (N*K)/6.5 \rfloor$, if (N*K)/6.5 is a real number with a decimal value < 0.5

$= \lceil (N*K)/6.5 \rceil$, if (N*K)/6.5 is a real number with a decimal value ≥ 0.5

The increased size of in_tabu list $= \lfloor (N*K)/4 \rfloor$, if (N*K)/4 is a real number with a decimal value < 0.5

$= \lceil (N*K)/4 \rceil$, if (N*K)/4 is a real number with a decimal value ≥ 0.5

where N = total number of bottleneck parts in the system.

K = the maximum number of all possible options for each part.

According to the formulae above, the in_tabu list sizes are dependent on the number of bottleneck parts (N) and the number of alternative options for each bottleneck part (K). Because there are only three options, K=3 for every bottleneck part. Thus, the in_tabu list sizes are truly dependent on the number of bottleneck parts.

The aspiration level/criterion for the inside search, called in_AL, is initially set equal to the total saving contributed by the inside initial solution. This list is updated as and when the total saving evaluated for the current parts' option configuration is found to be better than the total saving for the best parts' option configuration found so far.

(2) Inside candidate list (ICL) and inside index list (IIL)

The inside candidate list contains the potential configuration selected to perform future perturbations, while the inside index list consists of the local optima evaluated as the inside search progresses.

First, the inside initial solution FSp_0 is admitted into both ICL and IIL, and used as a initial node for perturbation. When all of the seeds have been evaluated for the initial node, the configuration that contributes to the highest objective function value (Z) is selected and admitted into the ICL and used as the new node for the next perturbation. The new configuration in ICL would receive a *star* if its objective function value (Z_1) is greater than its initial objective function value (Z_0). The star indicates that it has potential for becoming the next local optimum.

The new configuration FSp_1 is then perturbed in a similar fashion. The next configuration to be admitted into the ICL is selected as that having the highest objective function value (Z_2) from among the seeds perturbed from FSp_1. If $Z_2 \leq Z_1$, then the configuration corresponding to Z_1 would receive *double stars*, and would be admitted into the IIL as the first local optimum obtained for the inside search. Otherwise, Z_2 would receive a star. A configuration receiving a star has the potential for becoming the next local optimum. When a configuration receives double stars it is the next local optimum and, therefore, admitted into the IIL. The final solution for the inside search, indicating which option should be used for each bottleneck part, is selected as the configuration which gives the best total saving (Z) from among the local optimums identified (entries in the IIL).

(3) Number of iterations without improvement for the inside search (IN_INT)

The number of iterations for the inside search (in_iteration) is increased by one every time an in_move is performed. The number of iterations without improvement (IN_INT) is increased by if no improvement is found after an in_move is performed and reinitialized to zero whenever there is an improvement over the previous in_move.

The number of iterations without improvement is used as a stopping criterion to terminate the inside search. IN_INT is dependent on the size of the problem considered (i.e., the larger the problem, the larger is the IN_INT required to terminate the search). The number of iterations without improvement for the parts' option level is proportional to the number of bottleneck parts and the number of alternative options for each bottleneck part.

- For the fixed tabu list size, the inside stopping criterion is determined by the number of iterations without improvement for the inside search (IN_INT):

$$\text{IN\_INT} = \lfloor (N*K)/\text{reduction factor} \rfloor \quad \text{, if (N*K)/reduction factor is a real number with a decimal value} < 0.5$$

$$= \lceil (N*K)/\text{reduction factor} \rceil \quad \text{, if (N*K)/reduction factor is a real number with a decimal value} \geq 0.5$$

where N = total number of bottleneck parts

K = maximum number of options for each bottleneck part (always equals to 3)

reduction factor is assumed equal to 7 for the inside search, judged by the preliminary experimentation performed in this research.

- For the variable tabu list size, the inside stopping criteria are determined by;

(i) If there is no improvement within the last [int (IN_INT/3)] iterations with the initial in_tabu list size, then decrease the in_tabu list size to the decreased size evaluated in step 6.

(ii) If there is no improvement within the last [int (IN_INT/3)] iterations with the decreased size of in_tabu list, then increase the in_tabu list size to the increased size evaluated in step 6.

(iii) If there is no improvement within the last [int (IN_INT/3)] iterations with the increased size of in_tabu list, then stop the inside search and start diversifying.

Step 6x: To diversify the inside search performed in step 6, the mechanism called long-term memory has been implemented in the inside search.

The inside long-term memory (IN_LTM) is the frequency matrix that keeps track of the tenure of the bottleneck parts and its options. In other words, the IN_LTM will keep track of the number of times that each option has been used by each bottleneck part according to the history of solutions obtained for the inside search. The IN_LTM is updated regularly as the inside search progresses. Every time an in_move is performed, the entry in the frequency matrix (IN_LTM), which corresponds to the new part's option configuration is increased by one. By keeping track of the frequency of bottleneck part's options being used, the IN_LTM provides the information about which options are the most or least frequently used by each bottleneck part.

Using the information obtained from the frequency matrix, IN_LTM, the long-term memory based restart is generated. The restarts generate new initial configurations which are intended to diversify the search into new regions that were not previously investigated.

The new initial configuration is determined by the bottleneck parts' option configuration that was the best configuration found in the last restart. Two types of inside long-term memory are considered in this research: the inside long-term memory based on minimal frequencies (IN_LTM_MIN) and the inside long-term memory based on maximal frequencies (IN_LTM_MAX).

- The inside long-term memory based on maximal frequencies generates the restart by fixing the option for the bottleneck parts according to the maximal entry from the frequency matrix throughout the subsequent search.
- The inside long-term memory based on minimal frequencies generates the restart by fixing the option for the bottleneck parts according to the minimal entry from the frequency matrix throughout the subsequent search.

Once the restart configuration is obtained, reinitialize the in_tabu list and repeat the inside search (by performing step 4, 5, and 6) using this restart configuration as a new starting point until the required number of restarts for the inside search has been attained. The number of restarts required for the inside search is assumed equal to 2 in this research.

The minimal frequencies-based search will create new initial configurations in the new search regions that have not been investigated so far (diversifying the search). On the other hand, the maximal frequencies-based search will further explore in the regions considered "good" during the previous restart (intensifying the search). As the required number of restarts for the inside search has been reached, the inside search is terminated.

Step 7: When the inside search is terminated, the optimal/near-optimal bottleneck parts' option configuration would be determined as the one that contributes to the highest total saving found throughout the inside search. The direction of the search would be switched to the outside level.

Perform steps 3 through 6 (inside search) for each cell locations configuration outside, searching the seeds ($Sc(FSc_0)$) to evaluate the optimal/near-optimal bottleneck parts' option and the corresponding total saving. Every time an inside search is performed to obtain the new cell locations configuration in the seeds ($Sc(FSc_0)$, the parameters for the inside search that have to be reinitialized are in_tabu list, IN_INT, ICL, IIL, in_AL, and IN_LTM.

Step 8: Perform the tabu search, in the same fashion as the inside search, for the cell locations level (outside search). This process starts by moving from the initial cell locations configuration to the "best" candidate among its seeds. The out_move is identified by the move that transforms a cell locations configuration into another cell locations configuration considered among the seeds. By using the optimal/near-optimal total saving evaluated from the inside search for each cell locations configuration in the seeds, the out_move is performed in the same manner as the in_move. The value of the move and the aspiration criterion would also investigated in a similar fashion to those for the inside search. The following parameters for the outside tabu search are updated as the search progresses.

(1) Outside-tabu list (out_tabu list)

Every time an out_move is performed, the cell that is moved to the next adjacent location would be admitted into the out-tabu list along with its original location. The out_tabu list is updated circularly as the in_tabu list is updated in the inside search. Two types of out_tabu list are considered.

- The fixed tabu-list size for outside search is determined by the following formula.

The fixed size of out_tabu list $= \lfloor (C-1)/2 \rfloor$, if (C-1)/2 is a real number with a decimal value < 0.5

$= \lceil (C-1)/2 \rceil$, if (C-1)/2 is a real number with a decimal value ≥ 0.5

- The variable tabu-list sizes for outside search is determined by the following formulae.

The initial size of out_tabu list $= \lfloor (C-1)/2 \rfloor$, if (C-1)/2 is a real number with a decimal value < 0.5

$= \lceil (C-1)/2 \rceil$, if (C-1)/2 is a real number with a decimal value ≥ 0.5

The decreased size of out_tabu list $= \lfloor (C-1)/3 \rfloor$, if (C-1)/3 is a real number with a decimal value < 0.5

$= \lceil (C-1)/3 \rceil$, if (C-1)/3 is a real number with a decimal value ≥ 0.5

The increased size of out_tabu list $= \lfloor (C-1)/1.5 \rfloor$, if (C-1)/1.5 is a real number with a decimal value < 0.5

$= \lceil (C-1)/1.5 \rceil$, if (C-1)/1.5 is a real number with a decimal value ≥ 0.5

where C is the total number of cells.

For the perturbation of cell locations, the maximum number of seeds that can be generated is equal to (C-1) which means the out_move is limited to (C-1) alternatives. Realistically, therefore, the sizes of out_tabu list are proportional to (C-1) which is the number of seeds for each out_move.

Similar to the inside search, the aspiration criterion/level, namely out_AL, is created and initially set equal to the total saving for the initial cell locations configuration. The out_tabu status can be overwritten only when the corresponding cell locations configuration contributes to a total saving greater than the aspiration level at the current iteration.

(2) Outside candidate list (OCL) and outside index list (OIL)

An outside candidate list (OCL) and an outside index list (OIL) are created for the outside search in the same fashion as the inside search. OCL contains the potential cell locations configurations selected to perform future perturbation, while OIL consists of the local optima evaluated as the outside search progresses. The approaches used for admitting the cell locations configuration into the OCL and OIL are comparable to those for the ICL and IIL. Thus, the OCL and OIL are analogous to the ICL and IIL, respectively. The final configuration/solution, indicating which locations should be taken by each cell, is selected as the entry into the OIL which contributes the most total saving.

(3) <u>The Number of iterations without improvement for outside search (OUT_INT)</u>

The number of iterations without improvement for the outside search is created and updated similar to that for the inside search. The number of iterations without improvement for the outside search, namely OUT_INT, is increased by one if no improvement is found after moving from one cell locations configuration to another. Similar to step 7 for the inside search, the number of iterations without improvement, OUT_INT, would be used as a stopping criterion to terminate the outside search. The OUT_INT, used as the stopping criterion for the outside search, should be increased as the size of the problem considered became larger. The number of iterations without improvement for the outside search is evaluated as follows:

- For the fixed out_tabu list size, the number of iterations without improvement for the outside search (OUT_INT) is determined by:

$$\text{OUT\_INT} = \lfloor (C * N) / (\text{reduction factor} * M) \rfloor$$

where C = total number of cells

N = total number of bottleneck parts

M = total number of bottleneck machines, and

the reduction factor is assumed equal to 0.67 for the outside search, based on the preliminary experiment in this research.

- For the variable out_tabu list sizes; the outside search stopping criterion is determined by :

(i) If there is no improvement within the last [int (OUT_INT/3)] iterations with the initial out_tabu list size, then decrease the out_tabu list size to the decreased size given by $\lfloor (C-1)/3 \rfloor$, if (C-1)/3 is a real number with a decimal value < 0.5 or $\lceil (C-1)/3 \rceil$, if (C-1)/3 is a real number with a decimal value ≥ 0.5

(ii) If there is no improvement within the last [int (OUT_INT/3)] iterations with the decreased out_tabu list size, then increase the out_tabu list size to the increased size given by $\lfloor (C-1)/1.5 \rfloor$, if (C-1)/1.5 is a real number with a decimal value < 0.5 or $\lceil (C-1)/1.5 \rceil$, if (C-1)/1.5 is a real number with a decimal value ≥ 0.5

(iii) If there is no improvement within the last [int (OUT_INT/3)] iterations with the increased out_tabu list size, then stop performing the outside search.

Step 8x: To diversify the outside search the same mechanism, namely the long-term memory used with the inside search, is used. Outside long-term memory (OUT_LTM), comparable to IN_LTM, is the frequency matrix that keeps track of the tenure of cell locations. Similar to the IN_LTM, the OUT_LTM matrix is updated continuously as the outside search progresses. Whenever an out_move is performed to move a current cell locations configuration to a new cell locations configuration, the entries of the OUT_LTM that correspond to the new cell locations configuration are increased by one. By keeping track of this frequency matrix, the OUT_LTM provides the information about which specific location is most or least frequently occupied by each cell. The frequency entries in the OUT_LTM will also be used to construct the restarts for the outside search, in the same manner the entries in the IN_LTM are used in the inside search. Two types of outside long-term memory are considered in this research.

- OUT_LTM_MAX generates the restarts by fixing the position of the cell according to the maximal entry from the frequency matrix throughout the subsequent search.
- OUT_LTM_MIN generates the restarts by fixing the position of the cell according to the minimal entry from the frequency matrix throughout the subsequent search.

Once the restart configuration is obtained, the out_tabu list has to be reinitialized and the outside search repeated using this restart configuration as a new starting point recursively, until the required number of restarts for the outside search has been reached. In this research, the number of restarts for the outside search is assumed equal to 2.

The entire search would be terminated when the required number of restarts for the outside search (2) has been reached. The optimal/near-optimal cell locations configuration would be the one with the highest total saving evaluated throughout the search process. Moreover, the optimal/near-optimal cell locations configuration along with its optimal/near-optimal bottleneck parts' options configuration will be combined to obtain the final (optimal/near-optimal) solution for the original research problem.

5.4 APPLICATION OF THE HEURISTIC TO EXAMPLE PROBLEM

A simple example is considered to illustrate the functionality of the steps associated with the heuristic algorithm. The example illustrated here was derived from an example previously considered by Ramakrishna (1994) in the context of assessing the role of duplicating and subcontracting processes in the design of cellular manufacturing systems.

The original machine-part load matrix for this problem is presented in table B.1 (Appendix B). Table 5.1 presents the solution obtained for part-machine assignments with 3 cells, using the methodology proposed by Logendran (1991). For these assignment of parts and machines, there are a total of 5 bottleneck parts and 4 bottleneck machines as shown in Table 5.2. Also, the following assumptions have been made:

(i) Distance between any two cells = 1 unit

(ii) Cost for moving a unit load of a part by unit distance = $1.00

(iii) Size of unit load = 50

(iv) Amortized cost of bottleneck machines:

M1 = $700; M2 = $900; M3 = $1200; M4 = $1500

(v) Daily volume of production of bottleneck parts:

P1 = 365; P2 = 456; P3 = 321; P4 = 409; P5 = 487

(vi) Incremental cost of subcontracting the bottleneck parts:

$b_1 = 0.675$; $b_2 = 0.35$; $b_3 = 0.9$; $b_4 = 0.65$; $b_5 = 0.57$;

(vii) Average cost per unit of machine time:

R1 = $25; R2 = $27; R3 = $32; R4 = $39; R5 = $27;

R6 = $46; R7 = $41

(viii) Budgetary limit:

B = $500,000

(ix) Maximum number of machines that can be assigned to a cell

$N_l = 10$; $l = 1, 2, 3$

Table 5.1 Machines and part assignments for the three cells

Cell Number	Machine Assignments			Part Assignments
1	M7	M2	M6	P1, P4, P5, P6, P7, P9, P10
2	M4	M3		P3, P11, P12, P13
3	M1	M5		P2, P8, P14

Table 5.2 Bottleneck parts and bottleneck machines with respect to original part and machine numbers

Bottleneck Part #	Original Part #
1	5
2	7
3	8
4	9
5	10
Bottleneck Machine #	Original Machine #
1	1
2	3
3	4
4	6

Step 1: Generate the outside initial solution for alternative cell locations simply as: $FSc_0 = [1,2,3]$. It means that cell 1 is initially assigned to location 1, cell 2 is initially assigned to location 2, and cell 3 is initially assigned to location 3.

Step 2: Using $FSc_0 = [1,2,3]$ as a node, evaluate its seeds by perturbing on a location , yet one location at a time. As a result, the seeds of FSc_0 are evaluated as:

$Sc(FSc_0) = [2,1,3]$ and $[1,3,2]$

These are the only 2 seeds that can be evaluated according to the perturbation stated in Step 2 of the heuristic algorithm for this 3-cell example problem.

Step 3: Given the outside initial solution $FSc_0 = [1,2,3]$ as the cell locations configuration, evaluate the optimal/near-optimal total saving by performing the inside search. The inside search is initiated by evaluating the initial solution for bottleneck parts' options level. The following evaluations describe the procedure to obtain FSp_0 :

1). The savings contributed by duplicating bottleneck machines connected to each bottleneck part are evaluated as:

$DU_{5,1} = 8320$

$DU_{7,1} = 2600$

$DU_{8,3} = 3640$

$DU_{9,1} = 9360$

$DU_{10,1} = 5200$

2). The cost of duplicating the bottleneck machines connected each bottleneck part is evaluated as:

$E_{5,1} = 700$

$E_{7,1} = 1200$

$E_{8,3} = 1500$

$E_{9,1} = 1900$

$E_{10,1} = 900$

3). The savings contributed by subcontracting each bottleneck part is evaluated as:

$SC_{5,1} = 68850.6$

$SC_{7,1} = 25766$

$SC_{8,3} = 77768.6$

$SC_{9,1} = 77178.4$

$SC_{10.1} = 75888.8$

4). The incremental cost associated with subcontracting each bottleneck part is evaluated as:

$F_{5.1} = 64057.5$

$F_{7,1} = 41496$

$F_{8,3} = 75114$

$F_{9,1} = 69121$

$F_{10,1} = 72173.4$

5). The net savings due to duplicating the machine connected to each bottleneck part is evaluated as:

$NDU_{5,1} = 8320 - 700 = 7620$

$NDU_{7,1} = 2600 - 1200 = 1400$

$NDU_{8,3} = 3640 - 1500 = 2140$

$NDU_{9,1} = 9360 - 1900 = 7460$

$NDU_{10.1} = 5200 - 900 = 4300$

The net saving due to subcontracting each bottleneck part is evaluated as:

$NSC_{5,1} = 68850.6 - 64057.5 = 4793.1$

$NSC_{7,1} = 25766 - 41496 = -15730$

$NSC_{8,3} = 77768.6 - 75114 = 2654.6$

$NSC_{9,1} = 77178.4 - 69121 = 8057.4$

$NSC_{10,1} = 75888.8 - 72173.4 = 3715.4$

6). The maximum contribution due to each bottleneck part is evaluated as:

$MAXCON_{5,1} = \max [7620, 4793.1] = 7620$

$MAXCON_{7,1} = \max [1400, -15730] = 1400$

$MAXCON_{8,3} = \max [2140, 2654.6] = 2654.6$

$MAXCON_{9,1}$ = max [7460, 8057.4] = 8057.4

$MAXCON_{10,1}$ = max [4300, 3715.4] = 4300

From $MAXCON_{jl}$ above, the inside initial solution FSp_0 is evaluated as:

FSp_0 = [1,1,2,2,1]

Note: - 1 indicates option of machine duplication

- 2 indicates option of part subcontracting

- 0 indicates option of neither machine duplication nor part subcontracting

And the total savings for this initial solution is evaluated as:

TS = 7620 + 1400 + 2654.6 + 8057.4 + 4300 = 24032

7). Although the bottleneck parts 1, 2, and 5 in the parts' option configuration FSp_0 = [1,1,2,2,1] are assigned to the same cell 1, they are not connected to a common bottleneck machine. Thus, the adjustment term is evaluated as:

adj = 0

8). In the budgetary restriction, the total expense (E) for this initial bottleneck parts' option [1,1,2,2,1] is evaluated as:

The expense for bottleneck part 1 (due to duplication) = $E_{5,1}$ = 700

The expense for bottleneck part 2 (due to duplication) = $E_{7,1}$ = 1200

The expense for bottleneck part 3 (due to subcontracting) = $F_{8,3}$ = 75114

The expense for bottleneck part 4 (due to subcontracting) = $F_{9,1}$ = 69121

The expense for bottleneck part 5 (due to duplication) = $E_{10,1}$ = 900

The total expense (E) = 700 + 1200 + 75114 + 69121 + 900

= 147035

The budgetary limit B = \$500,000. As the expense (E) does not exceed the budget, the penalty for exceeding the budgetary limit is evaluated as:

Penalty = 0

9). To accommodate the constraint for the maximum number of machines that can be assigned to each cell (N_l), the total number of machines currently assigned to each cell (n_l) for the initial bottleneck parts' option [1,1,2,2,1] is evaluated as:

cell 1 $n_1 = 5$ machines $N_1 = 10$ machines

cell 2 $n_2 = 2$ machines $N_2 = 10$ machines

cell 3 $n_3 = 4$ machines $N_3 = 10$ machines

With $N_l = 10$ machines, the total number of machines in each cell currently does not exceed this limit. As a result, the penalty for exceeding the maximum number of machines that can be assigned to each cell is evaluated as:

Penalty 2 = 0

10). Taking into consideration of the adjustment, the budgetary restriction, and the limit on the number of machines that can be assigned to each cell, the objective function value (Z_{iint}) for the inside initial solution is evaluated as

$Z_{iint} = 24032 + 0 - 0 - 0 = 24032$

<u>Step 4</u>: Using the inside initial solution FSp_0 obtained in step 3 = [1,1,2,2,1] as a node, evaluate its seeds by perturbing on options for each bottleneck part, yet one part at a time. The seeds of FSp_0 are given by

$Sp(FSp_0) =$ [0,1,2,2,1], [2,1,2,2,1], [1,0,2,2,1], [1,2,2,2,1], [1,1,0,2,1], [1,1,1,2,1], [1,1,2,0,1], [1,1,2,1,1], [1,1,2,2,0], and [1,1,2,2,2].

<u>Step 5</u>: Evaluate the total savings (Z) for each seed obtained in step 4 by using the procedure outlined in step 3. For example, consider the seed $Sp(FSp_1)$ = [0,1,2,2,1]. The net saving contributed by each bottleneck part is

- $CON_{5,1} = 0$ because part 5 shows preference for neither duplicating option or subcontracting option.
- $CON_{7,1} = NDU_{7,1} = 1400$ because part 7 takes the duplicating option.
- $CON_{8,3} = NSC_{8,3} = 2654.6$ because part 8 takes the subcontracting option.
- $CON_{9,1} = NSC_{9,1} = 8057.4$ because part 9 takes the subcontracting option.

- $CON_{10,1} = NDU_{10,1} = 4300$ because part 10 takes the duplication option.

TS = 0 + 1400 + 2654.6 + 8057.4 + 4300 = 16412

adjustment = 0

E = 157047

penalty 1 = 0

The total number of machines currently assigned to cell 1 = 7 machines.

The total number of machines currently assigned to cell 2 = 2 machines.

The total number of machines currently assigned to cell 3 = 4 machines.

penalty 2 = 0

Total saving (Z_1) = 16412

Using the same approach, the total saving for each of the seeds obtained in step 5 is evaluated as

$Sp(FSp_1)$ = [0,1,2,2,1], the total saving Z_1 is equal to 16412

$Sp(FSp_2)$ = [2,1,2,2,1], the total saving Z_2 is equal to 21205.1

$Sp(FSp_3)$ = [1,0,2,2,1], the total saving Z_3 is equal to 22632

$Sp(FSp_4)$ = [1,2,2,2,1], the total saving Z_4 is equal to 6902.1

$Sp(FSp_5)$ = [1,1,0,2,1], the total saving Z_5 is equal to 21377.4

$Sp(FSp_6)$ = [1,1,1,2,1], the total saving Z_6 is equal to 23517.4

$Sp(FSp_7)$ = [1,1,2,0,1], the total saving Z_7 is equal to 15974.6

$Sp(FSp_8)$ = [1,1,2,1,1], the total saving Z_8 is equal to 25334.6

$Sp(FSp_9)$ = [1,1,2,2,0], the total saving Z_9 is equal to 19732

$Sp(FSp_{10})$ = [1,1,2,2,2], the total saving Z_{10} is equal to 23447.4

Step 6: Perform the inside search by considering the in_move. The in_move transforms a sequence of bottleneck parts' option considered for the initial solution into another sequence of bottleneck parts' option for one of its seeds. The value of in_move is evaluated by the total saving after the move - total saving before the move.

For example, the total saving for the initial feasible bottleneck parts' options configuration (Z_0) is \$24032. The first in_move would select [1,1,2,1,1] as the next configuration because it has the highest total savings from the configurations considered as seeds. Should there be two or more seeds which have the same value of move, the best-first strategy is used to break ties.

After performing the in_move, the following parameters have to be updated before the search is continued.

(1) Inside-tabu list (in_tabu list)

In the example, the first in_move is performed to move the initial feasible bottleneck parts' option configuration [1,1,2,2,1] to the next bottleneck parts' option configuration which is [1,1,2,1,1]. Noticeably, the fourth bottleneck part is the one that has been perturbed. Thus the bottleneck part (bottleneck part 4) along with its original option (2) are moved into the in_tabu list as the first element..

in_tabu list = [$p_4(2)$]

The interpretation of this entry in the in_tabu list is that option 2 has been chosen for bottleneck part 4 in the most recent iteration and it has been changed to another option (1 in this case).

The inside aspiration level (in_AL) is also updated when the total savings evaluated for the current feasible solution is higher than the best total savings found so far. For the first in_move, the total savings evaluated for the new configuration (25334.6) is higher than the total savings for the initial feasible solution (24032). Therefore, the in_AL is set equal to the total savings for the new configuration.

- in_AL = 25334.6

In addition, the in_tabu list size for this example is determined as follows:

- The fixed size of in_tabu list = $\lfloor (5*3)/5 \rfloor = 3$

- The variable sizes of in_tabu list

The initial size of in_tabu list = $\lfloor (5*3)/5 \rfloor = 3$

The decreased size of in_tabu list = $\lfloor (5*3)/6.5 \rfloor = 2$

The increased size of in_tabu list = $\lceil (5*3)/4 \rceil = 4$

(2) Inside candidate list (ICL) and inside index list (IIL)

As stated before, the initial bottleneck parts' option configuration is admitted into the ICL. The new configuration obtained for this example is also admitted into both ICL and IIL as it will be selected to perform future perturbations. Moreover, the new configuration has a better total savings (25334.6) than the total savings of the initial configuration (24032). Thus, it is given a star, to indicate that it has the potential of becoming the next local optimal.

- ICL = { [1,1,2,2,1], [1,1,2,1,1]* }
- IIL = { [1,1,2,2,1], [1,1,2,1,1] }

(3) Number of iterations without improvement for the inside search (IN_INT).

Every time an in_move is performed, the number of iterations for the inside search (in _iteration) is increased by one. If there is no improvement in total savings according to the recent in_move, the number of iteration without improvement for the inside search (IN_INT) is also increased by one. On the other hand, if for any in_iteration there is an improvement in total savings, the number of iteration without improvement for the inside search (IN_INT) will be reset to zero.

For this example, evidently there is an improvement according to the first in_move. Therefore, the number of iteration without improvement for the inside search (IN_INT) is reset to zero.

- in_iteration = 1
- IN_INT = 0

The number of iterations without improvement (IN_INT) is used as the stopping criterion to terminate the inside search which is determined as follows.

- For the fixed in_tabu list size, the inside stopping criterion is determined by the number of iterations without improvement for the inside search (IN_INT)

$IN\_INT = \lfloor (5*3)/7 \rfloor = 2$

- For the variable in_tabu list size, the inside stopping criteria are determined by:

(i) If there is no improvement within the last [int(IN_INT/3)] interactions with the initial in_tabu list size (3), then decrease the in_tabu list size to the decreased size of in_tabu list (2).

(ii) If there is no improvement within the last [int(IN_INT/3)] iterations with the decreased size of in_tabu list (2), then increase the in_tabu size to the increased size of in_tabu list size (4).

(iii) If there is no improvement within the last [int(IN_INT/3)] iterations with the increased size of in_tabu list (4), then stop the inside search.

The results for the inside search with fixed tabu-list size for FSc_0 = [1,2,3] using FSp_0 = [1,1,2,2,1] as an initial bottleneck parts' option configuration are shown in table 5.3

Table 5.3 Results obtained for the inside search of FSc_0 = [1,2,3], starting with FSp_0 = [1,1,2,2,1] as an initial bottleneck parts' option configuration.

# in_iteration	Entries into ICL	Total Savings (Z)	Entries into IIL
0	[1,1,2,2,1]**	24034	[1,1,2,2,1]
1	[1,1,2,1,1]**	25334.6	[1,1,2,1,1]
2	[1,1,1,1,1]	24820	
3	[1,1,1,1,2]	24235.4	

The inside search, starting with [1,1,2,2,1], is terminated after 3 iterations have been performed because the number of iterations without improvement for the fixed tabu-list size of 2 (IN_INT = 2) has been reached. The best configuration for the inside search is [1,1,2,1,1] which is also the first entry into the IIL with a highest total savings of $25334.6.

Step 6x: To diversify the inside search performed in step 6, the inside long-term memory is implemented. The inside long-term memory (IN_LTM) is the frequency matrix that keeps track of the tenure of an option for each bottleneck part throughout the inside search. Every time a new bottleneck parts' option configuration is constructed, the entries in IN_LTM matrix corresponding to the bottleneck parts and their respective options in the configuration are increased by one.

Originally, the entries in IN_LTM are all initialized to zero. After the first in_move, from initial bottleneck parts' option configuration [1,1,2,2,1] to the next configuration [1,1,2,1,1], is performed, the IN_LTM would be updated as shown in table 5.4.

Table 5.4 Updated IN_LTM frequency matrix after moving to the new configuration [1,1,2,1,1]

	Option 0 (Neither)	Option 1 (Duplicating)	Option 2 (Subcontracting)
Bottleneck Part 1	0	1	0
Bottleneck Part 2	0	1	0
Bottleneck Part 3	0	0	1
Bottleneck Part 4	0	1	0
Bottleneck Part 5	0	1	0

As the inside search progresses the IN_LTM frequency matrix is updated regularly. The corresponding IN_LTM frequency matrix for the inside search after the number of iterations without improvement (IN_INT) has been reached in step 6 is presented in Table 5.5

Table 5.5 The IN_LTM frequency matrix for the inside search of initial cell location configuration, $SFc_0 = [1,2,3]$, starting with $SFp_0 = [1,1,2,2,1]$ as an initial bottleneck parts' option configuration.

	Option 0 (Neither)	Option 1 (Duplicating)	Option 2 (Subcontracting)
Bottleneck Part 1	0	4	0
Bottleneck Part 2	0	4	0
Bottleneck Part 3	0	2	2
Bottleneck Part 4	0	3	1
Bottleneck Part 5	0	3	1

In order to use the long-term memory based on the maximal frequency for the inside search (IN_LTM_MAX), the next restart is activated by considering the maximal entry in the IN_LTM frequency matrix and fixing the bottleneck part and its respective option corresponding to this maximal entry.

For example, the maximal entry in the IN_LTM frequency matrix is equal to 4, and it corresponds to both option 1 of bottleneck part 1 and option 1 of bottleneck part 2. The row-wise first best strategy is used to break ties. Therefore, the maximal entry of 4 according to option 1 of bottleneck part 1 is used for generating the first new restart. The new initial configuration for the next restart is constructed from fixing the option of bottleneck part 1 to 1 and the other bottleneck parts remain at the same options as they were in the best configuration found in the last restart. As a result, the new initial configuration for the next restart is $[\underline{1},1,2,1,1]$. The underline indicates that option 1 for bottleneck part 1 is now fixed throughout the next restarted search. The search for the next restart would be performed in a similar fashion according to the procedure described in step 6. The results obtained with the first long-term memory restart and the resulting IN_LTM are shown in Table 5.6 and 5.7, respectively.

Using the same approach, the results obtained with the second long-term memory restart are presented in Table 5.8.

Table 5.6 Results obtained for the inside search of FSc_0 = [1,2,3], starting with FSp_0 = [1,1,2,1,1] as the inside first restart configuration.

# in_iteration	Entries into ICL	Total Savings (Z)	Entries into IIL
0	[1,1,2,1,1]**	25334.6	[1,1,2,1,1]
1	[1,1,2,1,2]	24750	
2	[1,1,2,2,2]	23447.4	

Table 5.7 The IN_LTM frequency matrix for the inside search of FSc_0 = [1,2,3], starting with FSp_0 = [1,1,2,1,1] as the inside first restart configuration.

	Option 0 (Neither)	Option 1 (Duplicating)	Option 2 (Subcontracting)
Bottleneck Part 1	0	3	0
Bottleneck Part 2	0	3	0
Bottleneck Part 3	0	0	3
Bottleneck Part 4	0	2	1
Bottleneck Part 5	0	1	2

When the number of restarts for the inside search (2) has been reached, the inside search would be terminated and the optimal/near-optimal bottleneck parts' option configuration would be selected as the one which contributes to the highest total savings form among the best solution obtained with each restart. Table 5.9 presents the best solution obtained with each restart for this example

Table 5.8 Results obtained for the inside search of FSc_0 = [1,2,3], starting with FSp_0 = [1,1,2,1,2] as the inside second restart configuration.

# in_iteration	Entries into ICL	Total Savings (Z)	Entries into IIL
0	[1,1,2,1,2]**	24750	[1,1,2,1,2]
1	[2,1,2,1,2]	21223.1	
2	[2,1,2,1,1]**	21807.7	[2,1,2,1,1]
3	[2,1,1,1,1]	21293.1	
4	[2,1,1,2,1]	20690.5	

Table 5.9 Summary of results obtained from the inside search of FSc_0 = [1,2,3] with two long-term memory restarts.

Number of Restart	The Best Solution in the IIL	Total Savings
Initial Restart	[1,1,2,1,1]	25334.6
First Restart	[1,1,2,1,1]	25334.6
Second Restart	[1,1,2,1,2]	24750

The optimal/near-optimal solution for this example is given by the bottleneck parts' option configuration of [1,1,2,1,1], contributing to a total savings of $25334.6. However, this being a simple example, the long-term memory based on maximal frequency did not improve upon the best solution obtained from the initial restart.

<u>Step 7</u>: Repeat steps 3 through 6 (inside search) for each cell locations configuration in the seeds obtained form step 2 (i.e., $FSc(FSp_0)$ = [2,1,3] and [1,3,2]). Table 5.10 shows the results for the inside search of each cell locations configuration in the seeds of [1,2,3].

Table 5.10 Results obtained for the inside search of each cell locations configuration in $FSc(FSp_0)$.

The Cell Location Configurations in the Seeds of [1,2,3]	The Optimal/Near-Optimal Bottleneck Parts' Option Configuration Obtained for the Inside Search	Corresponding Total Savings
[2,1,3]	[1,1,2,1,1]	19354.6
[1,3,2]	[1,1,2,1,1]	27154.6

Step 8: Similar to step 6 of the inside search, the out_move is now performed. The out_move transforms a sequence of cell locations configuration to another sequence of cell locations in its seeds Sc(FSc). The value of out_move and the aspiration criterion would also investigated in the same fashion as those for the inside search.

For this example, the out_move transforms the initial feasible cell locations configuration [1,2,3] to a new cell locations configuration [1,3,2] since it contributes to the highest total saving in its seeds.

Similar to the inside search (step 6), the following parameters for the outside search are also updated during the search.

(1) Outside-tabu list (out_tabu)

Consider the out_move in this example which moves the initial feasible cell locations configuration[1,2,3] to the next configuration [1,3,2]. The cell is moved to the next adjacent location would be admitted into the out_tabu list along with its original location. For the first out_move, cell 2 has been moved to the next adjacent location (moved from position 2 to position 3). Thus, cell 2 along with its location (2) would be moved into the out_tabu list as the first entry.

out_tabu = [$loc_2(2)$]

The interpretation of this entry in the out_tabu list is that cell 2 occupied location 2 in the most recent iteration and it has been moved to the next adjacent location (location

3). The out_tabu list is updated regularly as the in_tabu list for the inside search. Two types of out_tabu list are considered as well. The fixed tabu-list size and the variable tabu-list sizes are determined by the formulae stated previously. However, it is not appropriate to consider the variable tabu-list sizes because the number of cells is too small in this example.

- The fixed tabu-list size for the outside search is determined by the following formula.

The fixed size of out_tabu list = $\lfloor (3-1)/2 \rfloor = 1$

As for the inside search, the outside aspiration level (out_AL) is initially set equal to 25334.6 obtained for the initial cell locations configuration [1,2,3]. As the outside search progresses the out_AL is updated if the total saving evaluated for the current configuration is found to be better than the best configuration found so far. Thus, out_AL is updated to 27154.6 according to the new configuration [1,3,2].

Again, the out_tabu list forbids the search from moving to a configuration represented by the entries in the out_tabu list. However, the out_tabu status can be overwritten when the total saving evaluated for that configuration is better than the current aspiration level (out_AL).

(2) Outside candidate list (OCL) and outside index list (OIL):

Similar to the inside search, the initial feasible cell locations configuration is admitted into both OCL and OIL. The next configuration is also moved into the OCL as it will be considered to perform future perturbations. As this configuration contributes to a higher total saving compared to the initial configuration, it is also given a star because it has the potential of becoming the next local optimum.

- OCL = { [1,2,3], $[1,3,2]^*$ }
- OIL = { [1,2,3] }

(3) The number of iterations without improvement for the outside search

Similar to the inside search, the number of iterations without improvement for the outside search (OUT_INT) is increased by one, if there is no improvement in the total saving relative to the recent out_move. However, if in any iteration there is an improvement in total savings, the number of iterations without improvement will be reinitialized to zero. In this example, the first out_move shows an improvement in total savings (from 25334.6 to 27154.6). Thus, the number of iterations without improvement (OUT_INT) is set equal to zero.

The number of iterations without improvement is used as a stopping criterion to stop the outside search. The number of iterations without improvement for the outside search is determined by:

- For the fixed out_tabu list size (notice that only the fixed tabu-list size is considered in this example), the outside search stopping criterion is determined by the number of iterations without improvement (OUT_INT):

$$OUT\_INT = \lceil (3*5) / (0.67*4) \rceil = 6$$

The results obtained from performing the outside search is presented in table 5.11

Table 5.11 Results obtained for the outside search starting with FSc_0 = [1,2,3] as the initial cell location configuration

# out_iteration	Entries into OCL	Total Savings (Z)	Entries into OIL
0	[1,2,3]**	25334.6	[1,2,3]
1	[1,3,2]**	27154.6	[1,3,2]
2	[2,3,1]	19354.6	

The effect of cell locations in this example problem can be seen from the results presented in Table 5.11. Different cell locations configurations can have a significant impact on evaluating different maximum total savings. Therefore, taking cell location into consideration can be beneficial in determining the best solution for the entire system. However, this example has only 3 cells. The three different cell locations shown in Table 5.11 are the only distinguishable cell locations configurations. As a result the outside search in this example problem has been shortened. The use of long term-memory for the outside search is implemented a similar fashion as for the inside search. However, as the problem is small, the use of long-term memory did not improve the best solution obtained for the initial restart.

6. EXPERIMENTAL DESIGN

6.1 DETERMINATION OF THE OPTIMAL TOTAL SAVINGS

In the previous chapter, the optimal/near optimal total savings for the example problem was obtained with the tabu search-based heuristic using fixed tabu-list size and LTM_MAX (TSH 2). The same maximum total saving of $27154.6 was obtained with the remaining five of the six tabu search-based heuristic. The heuristics solution should be compared to the global optimal solution to determine how good or bad the solution is. An attempt to determine the optimal solution for the example problem has been made to compare the results with the solutions obtained from the tabu search based-heuristics. As the mathematical model for the research problem is a polynomial programming model, there is a possibility of determining the global optimal solution for small problem structures. This can only be accomplished by decomposing the polynomial programming model to give linear binary/general integer programming models where the cell locations are fixed. The illustration on how it is decomposed is provided in Appendix A.1 to A.3. The objective function of the decomposed problems would consist of linear terms. As such, SuperLINDO (1989) computer software can be used to optimally solve the decomposed problems. The maximum of the maximum objective function values obtained from the decomposed problems would also be the global optimal solution for the polynomial programming model, representing the original research problem.

The maximum number of different cell locations for this example problem with 3 cells is equal to 6 as shown in Figure 6.1. Of these 6 different cell locations, only 3 can be considered distinctively different. For example, configurations [1,2,3] and [3,2,1] representing pattern 1 have the same distances between any two cells. Thus, only 3 patterns will contribute to evaluating different total savings if alternative cell locations are considered. Therefore, each pattern is solved for the global optimal solution, using the SuperLINDO software. The global optimal solution for the original problem would be the pattern of cell locations which contribute the maximum total savings along with its best bottleneck part options configuration.

Figure 6.1 The three patterns of cell locations

Pattern 1	FSc = [1,2,3]	and	FSc = [3,2,1]
	C1 C2 C3		C3 C2 C1
Pattern 2	FSc = [2,1,3]	and	FSc = [2,3,1]
	C2 C1 C3		C3 C1 C2
Pattern 3	FSc = [1,3,2]	and	FSc = [3,1,2]
	C1 C3 C2		C2 C3 C1

In Table 6.1, the best solution obtained for each pattern with each of the six tabu search-based heuristics is compared to the optimal solution obtained with SuperLINDO software.

The best solution obtained for the small problem with tabu search-based heuristics matches with the optimal solution obtained with SuperLINDO (1989) software. This demonstrates that tabu search-based heuristics will have a high potential for finding good near-optimal solutions, if not optimal solutions, in medium and large problem structures. For the best solution [1,1,2,1,1] found here, the total expense is equal to $147035 and the total number of machines that has already been assigned to cells 1, 2, and 3 is equal to 8, 2, and 4, respectively. To further reinforce this observation, the comparison is extended to include three different cases which involve the budgetary limitation and the limitation on the maximum number of machines that can be assigned to each cell.

- Case 1 considers the situation when only the budget is limited to $50,000 (B = $50,000). The comparative results are presented in Table 6.2.

- Case 2 considers the situation when only the maximum number of machines that can be assigned to each cell is limited to 6 (Nc = 6). The comparative results are summarized in Table 6.3.

- Case 3 considers the situation when not only the budget is limited to $50,000 but also the maximum number of machines that can be assigned to each cell is limited to 6 (B = $50,000 and Nc = 6). The comparative results are presented in Table 6.4.

Table 6.1 Results obtained with the heuristics and the global optimum solution -Single-row layout.

Lindo		Tabu Search		
Pattern 1	Optimal Solution = [1,1,2,1,1]	Initial Solution	- Cell Location	= [1,2,3]
			- Best Solution	= [1,1,2,2,1]
	Total Saving = 25334.6		- Total Saving	= 24032
Pattern 2	Optimal Solution = [1,1,2,1,1]			
	Total Saving = 19354.6	Final Solution	- Cell Location	= [1,3,2]
Pattern 3	Optimal Solution = [1,1,2,1,1]		- Best Solution	= [1,1,2,1,1]
			- Total Saving	= 27154.6
	Total Saving = 27154.6			

For the example problem, the best solution obtained from tabu search-based heuristics matches with the optimal solution obtained from SuperLINDO (1989) software for the basic case as well as the three cases including the constraint on budgetary

Table 6.2 Results obtained with the heuristics and the global optimum solution (B = $50,000) - Single-row layout.

Lindo		Tabu Search		
Pattern 1	Optimal Solution = [1,1,1,1,1] Total Saving = 24820	Initial Solution	- Cell Location - Best Solution - Total Saving	= [1,2,3] = [1,1,1,1,1] = 24820
Pattern 2	Optimal Solution = [1,1,1,1,1] Total Saving = 18840	Final Solution	- Cell Location - Best Solution - Total Saving	= [1,3,2] = [1,1,1,1,1] = 26640
Pattern 3	Optimal Solution = [1,1,1,1,1] Total Saving = 26640			

Table 6.3 Results obtained with the heuristics and the global optimum solution (Nc = 6) - Single-row layout.

Lindo		Tabu Search		
Pattern 1	Optimal Solution = [1,0,2,2,2] Total Saving = 22047.4	Initial Solution	- Cell Location - Best Solution - Total Saving	= [1,2,3] = [1,0,2,2,2] = 22047.4
Pattern 2	Optimal Solution = [1,0,2,2,2] Total Saving = 16067.4	Final Solution	- Cell Location - Best Solution - Total Saving	= [1,3,2] = [2,1,2,2,2] = 22440.5
Pattern 3	Optimal Solution = [2,1,2,2,2] Total Saving = 22440.5			

Table 6.4 Results obtained with the heuristics and the global optimum solution (B = 50,000 with Nc = 6) - Single-row layout.

Lindo		Tabu Search		
Pattern 1	Optimal Solution =	Initial Solution	- Cell Location	= [1,2,3]
	[1,0,1,0,0]		- Best Solution	= [1,0,1,0,0]
	Total Saving = 9760		- Total Saving	= 9760
Pattern 2	Optimal Solution =			
	[0,0,1,0,1]			
	Total Saving = 4620	Final Solution	- Cell Location	= [1,3,2]
Pattern 3	Optimal Solution =		- Best Solution	= [0,0,1,0,1]
	[0,0,1,0,1]		- Total Saving	= 9820
	Total Saving = 9820			

limitation, maximum number of machines assigned to each cell, or both. It implies that the tabu search-based heuristics have a very high-potential for finding the optimal/near-optimal solution whether or not the constraints are binding.

The determination of the optimal solution for the double row layout is constructed in a similar fashion. The comparative results for including none, one, or both constraints for the double row layout arrangement are summarized in Tables 6.5, 6.6, 6.7, and 6.8, respectively.

As shown in Figure 6.2, the distance between any two cells is the same in both layout arrangements. This explains why the results obtained for the example problem with both layout arrangements are the same.

In conclusion, the tabu search-based heuristics have a very high potential for finding optimal or good near-optimal solutions for both single-row layout and double-row layout arrangements whether or not the constraints are binding. Thus, the use of tabu search-based heuristic to search for the optimal/near-optimal of medium and large problems is a valuable attempt.

Table 6.5 Results obtained with the heuristics and the global optimum solution (no constraints included) - Double-row layout.

<table>
<tr><td colspan="2">Lindo</td><td colspan="3">Tabu Search</td></tr>
<tr><td>Pattern 1</td><td>Optimal Solution =
[1,1,2,1,1]
Total Saving = 25334.6</td><td rowspan="2">Initial Solution</td><td rowspan="2">- Cell Location
- Best Solution
- Total Saving</td><td rowspan="2">= [1,2,3]
= [1,1,2,2,1]
= 24032</td></tr>
<tr><td>Pattern 2</td><td>Optimal Solution =
[1,1,2,1,1]
Total Saving = 19354.6</td></tr>
<tr><td>Pattern 3</td><td>Optimal Solution =
[1,1,2,1,1]
Total Saving = 27154.6</td><td>Final Solution</td><td>- Cell Location
- Best Solution
- Total Saving</td><td>= [1,3,2]
= [1,1,2,1,1]
= 27154.6</td></tr>
</table>

Table 6.6 Results obtained with the heuristics and the global optimum solution (B = $50,000) - Double-row layout.

<table>
<tr><td colspan="2">Lindo</td><td colspan="3">Tabu Search</td></tr>
<tr><td>Pattern 1</td><td>Optimal Solution =
[1,1,1,1,1]
Total Saving = 24820</td><td rowspan="2">Initial Solution</td><td rowspan="2">- Cell Location
- Best Solution
- Total Saving</td><td rowspan="2">= [1,2,3]
= [1,1,1,1,1]
= 24820</td></tr>
<tr><td>Pattern 2</td><td>Optimal Solution =
[1,1,1,1,1]
Total Saving = 18840</td></tr>
<tr><td>Pattern 3</td><td>Optimal Solution =
[1,1,1,1,1]
Total Saving = 26640</td><td>Final Solution</td><td>- Cell Location
- Best Solution
- Total Saving</td><td>= [1,3,2]
= [1,1,1,1,1]
= 26640</td></tr>
</table>

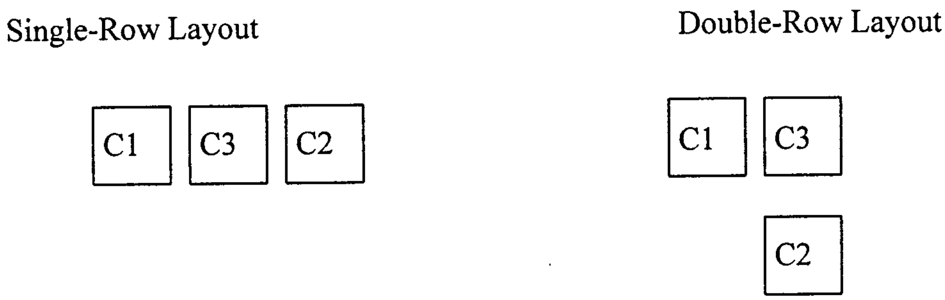
Table 6.7 Results obtained with the heuristics and the global optimum solution (Nc = 6) - Double-row layout.

Lindo		Tabu Search		
Pattern 1	Optimal Solution = [1,0,2,2,2] Total Saving = 22047.4	Initial Solution	- Cell Location - Best Solution - Total Saving	= [1,2,3] = [1,0,2,2,2] = 22047.4
Pattern 2	Optimal Solution = [1,0,2,2,2] Total Saving = 16067.4			
Pattern 3	Optimal Solution = [2,1,2,2,2] Total Saving = 22440.5	Final Solution	- Cell Location - Best Solution - Total Saving	= [1,3,2] = [2,1,2,2,2] = 22440.5

Table 6.8 Results obtained with the heuristics and the global optimum solution (B = 50,000 with Nc = 6) - Double-row layout.

Lindo		Tabu Search		
Pattern 1	Optimal Solution = [1,0,1,0,0] Total Saving = 9760	Initial Solution	- Cell Location - Best Solution - Total Saving	= [1,2,3] = [1,0,1,0,0] = 9760
Pattern 2	Optimal Solution = [0,0,1,0,1] Total Saving = 4620			
Pattern 3	Optimal Solution = [0,0,1,0,1] Total Saving = 9820	Final Solution	- Cell Location - Best Solution - Total Saving	= [1,3,2] = [0,0,1,0,1] = 9820

Figure 6.2 Single and double-row layouts for configuration [1,3,2]



The distance between C1 and C2 = 2 units in both layout arrangements.

6.2 COMPARISON OF TABU SEARCH-BASED HEURISTICS

The mechanisms that have a significant impact on tabu search-based heuristics are (i) using fixed versus variable tabu-list sizes, and (ii) using long-term memory versus not using it. To examine the effect of these mechanisms, six different tabu search heuristics have been constructed as shown in Table 6.9 and tested on different problem structures

Table 6.9 The six different tabu search-based heuristic algorithms

	Inside			Outside		
	tabu size	LTM_MIN	LTM_MAX	tabu size	LTM_MIN	LTM_MAX
TSH 1	const	-	-	const	-	-
TSH 2	const	Yes	-	const	Yes	-
TSH 3	const	-	Yes	const	-	Yes
TSH 4	var	-	-	var	-	-
TSH 5	var	Yes	-	var	Yes	-
TSH 6	var	-	Yes	var	-	Yes

As seen from Table 6.9, the tabu search-based heuristics that apply long term-memory for both inside and outside searches are TSH 2, 3, 5, and 6. TSH 2 & 5 are based on the maximum frequency (LTM_MAX), while TSH 3 & 6 are based on the minimum frequency (LTM_MIN). The variable tabu-list sizes in both inside and outside searches are implemented in TSH 4, 5, and 6

To compare the performance of the six different tabu search-based heuristics, a single-factor experiment is constructed. In this case, the factor is characterized by each of the different tabu search-based heuristic and measured by the highest total saving evaluated. As the test problems used with each heuristic can be different, the experiment is conducted as a randomized complete block design using the test problem as a block. Otherwise, the influence of differences in structure of the test problems can contribute to identifying a difference in the performance of the heuristics. Using the randomized complete block design the difference can be wholly attributed to the difference in performance of each heuristic itself, and not the difference between test problems. In this research each of the six heuristics is tested with a block (sample) size of 12, representing 12 different test problems.

The 12 different test problems are constructed to include none, one or both constraints which are the limitation on budget and the maximum number of machines that can be assigned to each cell. As a result, these problems can be categorized into four different classes:-

- One out of the 12 problems with no limitation on budget or the maximum number of machines that can be assigned to each cell.
- Three out of the 12 problems include the limitation on budget but do not include the limitation on maximum number of machines that can be assigned to each cell.
- Two out of the 12 problems include the limitation on the maximum number of machines that can be assigned to each cells but do not include the limitation on budget.
- Six out of the 12 problems include both constraints which probably is more meaningful when compared to an actual manufacturing system.

Test problems are generated randomly to exhibit the features corresponding to these 4 classes. For more details on randomized block designs, the reader is advised to refer to the text by Montgomery (1991).

Three different problem structures are tested in this research. Each problem structure is defined by the number of cells, the number of bottleneck parts, and the number of bottleneck machines in the cellular manufacturing system. The first problem structure is similar to the example problem considered in Section 5.5 which consists of three cells (C), five bottleneck parts (P), and four bottleneck machines (M), denoted by 3C*5P*4M. The machine-part load matrix for the first problem structure is shown in Table B.1 (Appendix B). The same problem structure was previously used by Ramakrishna (1994). The second , considered as the medium-size problem structure, is denoted by 5C*13P*8M. The machine-part load matrix for this problem structure is derived from published literature (Seifoddini, 1989), and is presented in Table B.3 (Appendix B). The third, considered as the large-size problem structure, is denoted by 8C*27P*21M. The machine-part load matrix for this problem structure is also derived from the published literature (Burke and Kamal, 1995), and is shown in Table B.5 (Appendix B).

In addition, the parameters used with the tabu search-based heuristics for each problem structure are given in Table B.7 (Appendix B). The data for these parameters are generated randomly from uniform distributions for each problem structure. These are as follows: Amortized cost of bottleneck machines from [3000,6000], daily volume of production of bottleneck parts from [300,800], incremental cost of subcontracting the bottleneck parts from [0.2,0.6], and the average cost per unit of machining time from [30,60]. The randomly generated data used with 3C*5P*4M, 5C*13P*8M, and 8C*27P*21M problem structures are shown in Tables B.2, B.4, and B.6 , respectively.

For each problem structure, 12 different blocks (test problems) are generated and the maximum total saving for each block with each of the six heuristics is determined. An analysis of variance is performed to determine if the average total savings obtained for those 12 problems is significantly different between the six heuristics. In this experiment, the significance level α, also referred to as type I error, is assumed equal to

5% (α=0.05). When a difference in the average total saving is found, a Least Significance Difference (LSD) test is performed to identify which heuristics contributed to the difference. In this research LSD is selected instead of other tests such as Duncan's Multiple Range, Newman Keul's and Tukey's, because it is available in Excutstat (Version 3.0), a computerized statistical programming software package.

7. RESULTS AND DISCUSSIONS

The experimental results for each test problem obtained from applying each heuristic algorithm along with the CPU time are illustrated in Table C.1- C.6 (Appendix C), for the 3C*5P*4M (single row layout), 5C*13P*8M (single row layout), 8C*27P*21M (single row layout), 3C*5P*4M (double row layout), 5C*13P*8M (double row layout), and 8C*27P*21M(double row layout) problem structures, respectively. Also, the results from the analysis of variance along with the LSD analysis for each layout arrangement and each problem structure are presented in Table D.1- D.6 (Appendix D).

The summary of the results above for the total savings along with the LSD analysis for each problem structure are shown in Table 7.1 and Table 7.2 for single row layout and double row layout, respectively. Furthermore, the results obtained from the LSD analysis are summarized in terms of the homogeneous group for each problem structure as presented in Table 7.3-7.5 and Table 7.6-7.8 for single row layout and double row layout, respectively. The "X" used in these tables denote the heuristics that do not differ significantly based on the LSD analysis.

Consider the single row layout arrangement. The results presented in Table 7.1 indicate that there is no significant difference among the six heuristics at $\alpha = 0.05$ for every problem structure tested. For the small size, 3C*5P*4M problem structure, TSH 1-6 determine the exact same maximum total saving of $92047.83 as shown in Table 7.3.

The medium size, 5C*13P*8M problem structure, also does not indicate a significant difference among the six heuristics as seen from the results presented in Table 7.4. However, TSH 2 & 5 determined a better maximum total savings of $299633 than other TSHs' which determined a total savings of $299343. The percentage difference is only 0.0968 % which is small enough to ignore the difference between these two groups of TSHs.

In Table 7.5, the results of 8C*27P*21M problem structure do not indicate a significant difference between the TSHs with $\alpha = 0.05$, even though there is a numerical

Table 7.1 Summary of results obtained for the comparison of TSH 1-TSH 6 for single row layout.

Average Total Savings (Z) with Number of Blocks =12	Problem Structure		
	3C*5P*4M	5P*13P*8M	8C*27P*21M
TSH 1	92047.83	299343	1405560
TSH 2	92047.83	299663	1406870
TSH 3	92047.83	299343	1406950
TSH 4	92047.83	299343	1404370
TSH 5	92047.83	299663	1407260
TSH 6	92047.83	299343	1407670
Is Z significant Different between TSH at α 0.05?	No	No	No
TSH 1 vs TSH 2	-	No	No
TSH 1 vs TSH 3	-	No	No
TSH 1 vs TSH 4	-	No	No
TSH 1 vs TSH 5	-	No	No
TSH 1 vs TSH 6	-	No	No
TSH 2 vs TSH 3	-	No	No
TSH 2 vs TSH 4	-	No	No
TSH 2 vs TSH 5	-	No	No
TSH 2 vs TSH 6	-	No	No
TSH 3 vs TSH 4	-	No	No
TSH 3 vs TSH 5	-	No	No
TSH 3 vs TSH 6	-	No	No
TSH 4 vs TSH 5	-	No	No
TSH 4 vs TSH 6	-	No	No
TSH 5 vs TSH 6	-	No	No

Table 7.2 Summary of results obtained for the comparison of TSH 1-TSH 6 for double row layout.

Average Total Savings (Z) with Number of Blocks =12	Problem Structure		
	3C*5P*4M	5P*13P*8M	8C*27P*21M
TSH 1	92047.83	237559	952839
TSH 2	92047.83	235971	1041730
TSH 3	92047.83	237260	1030070
TSH 4	92047.83	242256	1018240
TSH 5	92047.83	229500	932038
TSH 6	92047.83	229465	925261
Is Z significant Different between TSH at α 0.05?	No	Yes	Yes
TSH 1 vs TSH 2	-	No	Yes
TSH 1 vs TSH 3	-	No	No
TSH 1 vs TSH 4	-	Yes	No
TSH 1 vs TSH 5	-	Yes	No
TSH 1 vs TSH 6	-	Yes	No
TSH 2 vs TSH 3	-	No	No
TSH 2 vs TSH 4	-	Yes	No
TSH 2 vs TSH 5	-	Yes	Yes
TSH 2 vs TSH 6	-	Yes	Yes
TSH 3 vs TSH 4	-	Yes	No
TSH 3 vs TSH 5	-	Yes	Yes
TSH 3 vs TSH 6	-	Yes	Yes
TSH 4 vs TSH 5	-	Yes	No
TSH 4 vs TSH 6	-	Yes	Yes
TSH 5 vs TSH 6	-	No	No

difference in the total savings obtained with each TSH. TSH 6 determined the best total savings followed by TSH 5, TSH 3, TSH 2, TSH 1, and TSH 4.

Table 7.3 The LSD analysis of the results obtained from 3C*5P*4M (single row layout) problem structure in term of Homogeneous Group

Heuristics	Average Total Savings	Homogeneous Group
TSH 1	92047.83	X
TSH 2	92047.83	X
TSH 3	92047.83	X
TSH 4	92047.83	X
TSH 5	92047.83	X
TSH 6	92047.83	X

Table 7.4 The LSD analysis of the results obtained from 5C*13P*8M (single row layout) problem structure in term of Homogeneous Group

Heuristics	Average Total Savings	Homogeneous Group
TSH 2	299663	X
TSH 5	299663	X
TSH 6	299343	X
TSH 3	299343	X
TSH 1	299343	X
TSH 4	299343	X

On the other hand, the results from double row layout arrangement determined a significant difference between the TSHs on the 5C*13P*8M and 8C*27P*21M problem structures. For the 3C*5P*4M small-size problem structure, TSH 1-6 evaluate the exact same maximum total savings of $92047.83 as presented in Table 7.6.

For the 5C*13P*8M problem structure shown in Table 7.7, TSH 4 performed the best with an average total savings of $242256. The next homogeneous group consists of

TSH 1, TSH 3, and TSH 2 which evaluate an average total savings of $237559, $237560, and $235971, respectively. In contrast, both TSH 5 and TSH 6 evaluate "inferior" total savings compared to the other homogeneous groups.

Table 7.5 The LSD analysis of the results obtained from 8C*27P*21M (single row layout) problem structure in term of Homogeneous Group

Heuristics	Average Total Savings	Homogeneous Group
TSH 6	1407670	X
TSH 5	1407260	X
TSH 3	1406950	X
TSH 2	1406870	X
TSH 1	1405560	X
TSH 4	1404370	X

Table 7.6 The LSD analysis of the results obtained from 3C*5P*4M (double row layout) problem structure in term of Homogeneous Group

Heuristics	Average Total Savings	Homogeneous Group
TSH 1	92047.83	X
TSH 2	92047.83	X
TSH 3	92047.83	X
TSH 4	92047.83	X
TSH 5	92047.83	X
TSH 6	92047.83	X

Although there are 4 different homogeneous groups among the TSHs in the 8C*27P*21M problem structure, the best homogenous group consists of TSH 2, 3, and 4, which evaluate the maximum total savings of $1041730, $1030070, and $101824, respectively. The total savings of $932038 and $925261 evaluated with TSH 5 and TSH

6, respectively are noticeably worse than the total savings evaluated with TSH 2, TSH 3, TSH 4, and TSH 1. The same was true for the 5C*13P*8M problem structure.

Table 7.7 The LSD analysis of the results obtained from 5C*13P*8M (double row layout) problem structure in term of Homogeneous Group

Heuristics	Average Total Savings	Homogeneous Group		
TSH 4	242256	X		
TSH 1	237559		X	
TSH 3	237260		X	
TSH 2	235971		X	
TSH 5	229500			X
TSH 6	229465			X

Finally, the use of long-term memory and variable tabu list sizes in tabu search-based heuristics can be described as follows:

In the single row layout arrangement, TSH 2, 3, 5, and 6 which use the long-term memory have consistently determined a better maximum total savings than TSH 1 and 4 which did not use the long-term memory. This is true on all problem structures with the exception of the small problem structure (3C*5P*4M). When the size of the problem becomes larger, the difference in performance of TSH 2, 3, 5, and 6 is more pronounced than TSH 1 and 4 as seen from the better total savings evaluated in Table 7.5. For the comparison of the use of long-term memory based on maximal frequency (LTM_MAX) with the use of long term-memory based on minimal frequency (LTM_MIN), the heuristics using LTM_MIN (TSH 3 and 6) have determined the better average total savings than the heuristics using LTM_MAX (TSH 2 and 5) in the 8C*27P*21M problem structure. However, for the 5C*13P*8M problem structure, LTM_MAX performed better than LTM_MIN, but only with a negligible percentage difference. Thus, in general, the use of long term-memory based on minimum frequency (LTM_MIN) is more efficient than the use of long term-memory based on maximum frequency (LTM_MAX). The use of variable tabu-list sizes determined a better maximum total savings only when combined with the use of long-term memory. From the Table 7.5, the average maximum total savings with TSH 6 is better than TSH 3 and that with TSH 5 is better than TSH 2. In contrast, the use of variable tabu-list size in itself (TSH 4) determined an inferior solution than TSH 1.

With the double row layout arrangement, the combined use of long-term memory and variable tabu-list sizes in TSH 5 and TSH 6 clearly determined solutions inferior to the rest of the heuristics (TSH 1, 2, 3, and 4). Again, this is true for all problem structures except the small problem structure (3C*5P*4M). When long term-memory is not considered, TSH 4 which uses the variable tabu-list size has consistently determined a better average maximum total savings than TSH 1 which did not use the variable tabu list-size as seen from the results presented in Table 7.7 and 7.8. Clearly, TSH 4 has outperformed the other heuristics in terms of average total savings for the 5C*13P*8M problem structure. For the 8C*27P*21M problem structure, although TSH 2 and 3 determined a better average total savings than TSH 4, in a statistical sense TSH 2, 3, and 4 all belong to the same homogeneous group. Thus, in general, the use of variable tabu-list size and no long-term memory (TSH 4) is more efficient to search for the maximum total savings in the double-row layout arrangement.

Table 7.8 The LSD analysis of the results obtained from 8C*27P*21M (double row layout) problem structure in term of Homogeneous Group

Heuristics	Average Total Savings	Homogeneous Group			
TSH 2	1041730	X			
TSH 3	1030070	X	X		
TSH 4	1018240	X	X	X	
TSH 1	952839		X	X	X
TSH 5	932038			X	X
TSH 6	925261				X

In conclusion, it can be stated that for the single-row layout arrangement TSH 6, characterized by the tabu search-based heuristic with the use of long-term memory based on minimal frequency (LTM_MIN) and constant tabu-list size, has high potential to outperform the other heuristics. Therefore, TSH 6 is recommended for solving the problem considered in this research. For the double row layout, in two out of three problem structures, TSH 5 and 6 were found inferior to the rest of the heuristics. Furthermore ,TSH 4, incorporating the use of variable tabu list-sizes and no long term memory, was found to be the efficient heuristic for solving the problem considered in this research. Thus, for the double-row layout arrangement, TSH 4 is recommended.

8. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

The problem of simultaneously dealing with duplicating bottleneck machines and subcontracting bottleneck parts is investigated in the presence of alternative cell locations. The model for this problem is formulated as a polynomial programming model and is proven to be NP-hard in the strong sense. This rules out the possibility of employing an implicit enumeration-based technique to determine the optimal solution even on problems with moderate number of bottleneck parts and bottleneck machines. A higher-level heuristic, based on a concept known as tabu-search, is proposed to efficiently solve the problem.

Six different versions of the tabu search-based heuristic algorithm are tested on three different problem structures and two different layout arrangements. An extensive statistical analysis based on a randomized-block design has been performed to compare the performance of six heuristics (TSH 1 - TSH 6) using maximum total savings as the criterion. For the single- row layout arrangement, TSH 6, the tabu search-based heuristic using long term-memory based on minimal frequency (LTM_MIN) and constant tabu list size, is recommended. However, for the double-row layout arrangement, TSH 5 & 6 were found inferior to other heuristics. Therefore, TSH 4, characterized by the use of variable tabu-list sizes and no long term-memory, is recommended for solving this problem.

Further research can be performed by taking into consideration of other important practical design constraints (Heragu 1994).

In this research, the limit on the number of machines assigned to each cell includes the machines originally assigned and those that are duplicated. Realistically, this can be changed due to technological and safety considerations. Technical considerations may dictate two or more machines to be placed in the same cell to avoid redundancy. A good example of this is the heat-treatment station. Conversely, two or more work stations cannot be placed in the same cell because of safety considerations, such as painting and welding work stations. These work stations should be located in different cells or as far as possible because there may be a high interaction between them. Future research can be performed by including these special constraints in the model to evaluate more meaningful solutions to the problem.

In the evaluation of material handling costs, only the inter-cell moves are considered in this research. In practice, however, there are some huge cellular manufacturing systems where the material handling cost contributed by inter-cell moves quite significant that it can not be disregarded. Future research may also be performed by including intra-cell moves in the model to determine the effect of cell locations in the processes of machine duplication and part subcontracting.

BIBLIOGRAPHY

Askin, R.G., and Chui, K.S., 1990, A graph partitioning procedure for machine assignment and cell formation in group technology, *International Journal of Production Research*, 28: 1555-1572.

Askin, R. G., Creswell, S. H., Goldberg, J. B., and Vakharia, A. J., 1991, A hamiltonian path approach to reordering the part-machine matrix for cellular manufacturing, *International Journal of Production Research*, 29: 1081-1100.

Atmani, A., Lashkari, R. S., and Caron, R. J., 1995, A mathematical programming approach to joint cell formation and operation allocation in cellular manufacturing, *International Journal of Production Research*, 33: 1-15.

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.

Burbidge, J. L., 1971, Production flow analysis, *Production Engineer*, 50: 139-152.

Burbidge, J. L., 1973, AIDA and group technology, *International of Production Research*, 11: 315-324.

Burbidge, J. L., 1977, A manual method for production flow analysis, *Production Engineer*, 56: 34-38.

Burke, L., and Kamal, S., 1995, Neural network and the part family/machine group formation problem in cellular manufacturing: a framework using fuzzy ART, *Journal of Manufacturing Systems*, 14: 148-159.

Carrie, A. S., 1973, Numerical taxonomy applied to GT. 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: 64-76.

Chandrasekaran, M. P., and Rajagopalan, R., 1986a, An ideal seed on hierarchical clustering algorithm for cellular manufacturing, *International Journal of Production Research*, 24: 451-464.

Chandrasekaran, M. P., and Rajagopalan, R, 1986b, MODROC : an extension of rank order clustering of group technology, *International Journal of Production Research*, 24:1221-1233.

Chandrasekaran, M. P., and Rajagopalan, R, 1987, ZODIAC: an algorithm for concurrent formation of part families and machine cells, *International Journal of Production Research*, 25: 835-850.

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.

Choobineh, F., 1988, A framework for the design of cellular manufacturing systems, *International Journal of Production Research*, 26: 1161-1172.

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

De Beer, C., and De Witte, J., 1978, Production flow synthesis, *Annals of CIRP*, 27: 341-389.

De Beer, C., Van Gerwen, R., and De Witte, J., 1976, Analysis of engineering production systems as a base for production oriented reconstruction, *Annals of CIRP*, 25: 439-441.

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

Execustat Version 3.0, 1991(Strategy Plus Inc.)

Faber, Z., and Carter, M. W., 1986, A new graph theoretic approach for forming machine cells in cellular production systems, *Flexible Manufacturing Systems: Methods and Studies*, Kusiak, A., ed. New York: North Holland: 301-318.

Gongaware, T., and Ham, I., 1981, Cluster Analysis Applications for group technology manufacturing systems, *Ninth Amer. Meter Working Res. Conf.*, Society of Manufacturing Engineers.

Han, C., and Ham, I., 1986, Multiobjective cluster analysis for part family formations, *Journal of Manufacturing Systems*, 5: 223-230.

Harhalakis, G., Nagi, R., and Proth, J. M., 1990, An Efficient heuristic in manufacturing cell formation for group technology application, *International Journal of Production Research*, 28: 185-198.

Heragu, S. S., 1994, Group technology and Cellular Manufacturing, *IEEE Transactions on Systems, Man, and Cybernetics*, 24: 203-215.

Irani, S. A., Cohen, P. H., and Cavaliar, T. M., 1992, Design of Cellular Manufacturing Systems, *Transaction of the ASME*, 114: 352-360.

Jordan, P.C. and Frazier, G.V., 1993, Is the Full Potential of Cellular Manufacturing being Achieved?, *Production and Inventory Management Journal*, 34(1): 70-72.

Kang, S. L., and Wemmerlov, U., 1993, A work load-oriented heuristic methodology for manufacturing cell formation allowing reallocation of operations, *European Journal of Operational Research*, 69: 292-311.

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

King, J. R., 1980b, Machine-component group formation in group technology, *OMEGA*, 8: 193-199.

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.

Kini, R. B., Taube, L. R., and Mosier, C. T., 1991, Part identification and group technology : a new approach, *Journal of Manufacturing Systems*, 10:134-145.

Kumar, K. R., Kuisak, A., and Vannelli, A., 1986, Grouping of parts and components in flexible manufacturing systems, *European Journal of Operation Research*, 24: 387-397.

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

Kusiak, A., and Chow, W. S., 1988, Decomposition of manufacturing systems, *IEEE Trans. Robotics and Automation*, 4: 457-471.

Kusiak, A., and Chung, Y. K., 1991, GT/ART: using neutral network to from machine cells, *Manufacturing Review*, 4: 293-301.

Logendran, R., 1990, A workload based model for minimizing total intercell and intracell moves in cellular manufacturing, *International Journal of Production Research*, 28(5): 913-925.

Logendran, R., 1991, Impact of Sequence of Operations and Layout of Cells in Cellular Manufacturing, *International Journal of Production Research*, 29(2): 375-390.

Logendran, R., 1992, Effect of Alternative Process Plans in Manufacturing Cell Design: Complexity Issues, Proceedings, *18th Annual NSF Grantees Conference on Design and Manufacturing Systems Research*, Atlanta, GA, January 8-10: 885-889.

Logendran, R., 1993, A Three Phase Methodology for the Design of Cellular Manufacturing Systems, Proceedings, *19th Annual NSF Grantees Conference on Design and Manufacturing Systems Research, Charlotte, NC*, January 6-8: 1125-1128

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(3): 675-693.

Logendran, R., Ramakrishna, P., and Sriskandarajah, C., 1992, Design of Manufacturing Cells Via Tabu Search. Proceedings, *Second International Conference on Flexible Automation and Information Management (FAIM'92), Falls Chaarch, VA*: 633-644.

Logendran, R., Ramakrishana, P., and Sriskandarajah, C., 1994, Tabu Search Based Heuristics for Cellular Manufacturing Systems in the Presence of Alternative Process Plans, *International Journal of Production Research*, 32(2): 273-297.

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

McComick, W. T., Schweitzer, P. J., and White, T. W., 1972, Problem decomposition and data recognition by clustering technique, *Journal of Operation Research*, 20: 992-1009.

Offodile, O. F., 1991, Application of similarity coefficient method to part coding and classification analysis in group technology, *Journal of Manufacturing Systems*, 10: 442-448.

Offodile, O. F., Mehrez, A., and Grznar, J., 1994, Cellular manufacturing: a taxonomic review framework, *Journal of Manufacturing Systems*, 13: 196-220.

Rajagopalan, R., and Batra, J. L., 1975, Design of cellular production systems-a graph theoretic approach, *International Journal of Production Research*, 13: 567-579.

Rajamani, D., Singh, N., and Aneja, Y. P., 1990, Integrated design of cellular manufacturing systems in the presence of alternative process plan, *International Journal of Production Research*, 28(8) :1541-1554.

Ramakrishna, P., 1994, The role of duplicating and subcontracting processes in the design of cellular manufacturing systems, *MS Thesis # 30805-136*, Oregon State University, OR 1036, 1994.

Sakaran, S., and Kasilingam, R. G., 1993, On cell size and machine requirement planning in group technology systems, *European Journal of Operational Research*, 69: * 373-383.

Skinner, W., 1974, The focused factory, *Harvard Business Review*, May-June:113-121.

Seifoddini, H., 1989, Duplication process in machine cells formulation in group technology, *IIE Transactions*, 21: 382-388.

Seifoddini, H., and Djassemi, M., 1995, Merits of the production volume based similarity coefficient in machine cell formation, *Journal of Manufacturing Systems*, 14, 35-45.

Seifoddini, H., and Wolfe, P. M., 1987, Application of the similarity coefficient method n group technology, *IIE Transactions*, 18: 271-277.

Srinivasan, G., and Narendran, T. T., 1991, GRAFICS-a nonhierachies clustering algorithm for group technology, *International Journal of Production Research*, 29: 463-478.

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.

SuperLINDO,1989 (Chicago, IL: Lindo System, Inc.)

Tabucanon, M. T., and Ojha, R., 1987, ICRMA - A heuristic approach for intracell flow reduction in cellular manufacturing, *Material Flow*, 4: 187-197.

Vakharia, A. J., and Wemmerlov, U., 1990, Designing a cellular manufacturing systems: a material flow approach based on operation sequence, *IIE Transactions*, 22: 84-97.

Vanelli, A., and Kumar, K. R., 1985, A method for finding minimal bottle-neck cells for grouping part-machine families, *International Journal of Production Research*, 24: 387-400.

Vanelli, A., and Kumar, K. R., 1987, Strategic subcontracting for efficient desegregated manufacturing, *International Journal of Production Research*, 25: 1715-1728.

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

Wemmerlov, U., and Hyer, N. L., 1986, Procedures for the part-family machine group identification problem in cellular manufacturing, *Journal of Management*, 6: 125-147.

APPENDICES

APPENDIX A.

APPENDIX A.1

Pattern 1: Mathematical Formulation for the Example Problem in SuperLINDO

```
MAX    8320 X151 + 2600 X471 + 4680 X491 + 4680 X191 + 5200 X3101
   + 3640 X683 + 4793.1 Y51 - 15730 Y71 + 8057.4 Y91 + 6247.8 Y101
   + 2654.6 Y83 - 1200 R41 - 700 R11 - 900 R31 - 1500 R63
 SUBJECT TO
     2)  X3101 + R41 + R11 <=  5
     3)  R63 <=  6
     4)  64057.5 Y51 + 41496 Y71 + 69121 Y91 + 69641 Y101 + 75114 Y83
   + 1200 R41 + 700 R11 + 900 R31 + 1500 R63 <=  500000
     5)  2.5 X3101 - 8 R31 <=  0
     6)  1.35 X471 + 1.03 X491 - 8 R41 <=  0
     7)  2.42 X151 + 2.48 X191 - 8 R11 <=  0
     8)  2.26 X683 - 8 R63 <=  0
     9)  Y51 + Z51 <=  1
    10)  Y71 + Z71 <=  1
    11)  Y91 + Z91 <=  1
    12)  Y101 + Z101 <=  1
    13)  Y83 + Z83 <=  1
    14)  X151 - Z51 =  0
    15)  X471 - Z71 =  0
    16)  X491 + X191 - 2 Z91 =  0
    17)  X3101 - Z101 =  0
    18)  X683 - Z83 =  0
END
INTE   X151
INTE   X471
INTE   X491
INTE   X191
INTE   X3101
INTE   X683
INTE   Y51
INTE   Y71
INTE   Y91
INTE   Y101
INTE   Y83
INTE   Z51
INTE   Z71
INTE   Z91
INTE   Z101
INTE   Z83
GIN    R41
GIN    R11
GIN    R31
GIN    R63
```

APPENDIX A.2

Pattern 2: Mathematical Formulation for the Example Problem in SuperLINDO

```
MAX    4160 X151 + 5200 X471 + 3020 X491 + 2340 X191 + 10400 X3101
   + 1820 X683 + 633.1 Y51 - 13130 Y71 + 8057.4 Y91 + 11447.8 Y101
   + 834.6 Y83 - 1200 R41 - 700 R11 - 900 R31 - 1500 R63
SUBJECT TO
    2)  X3101 + R41 + R11 <=  5
    3)  R63 <=  6
    4)  64057.5 Y51 + 41496 Y71 + 69121 Y91 + 69641 Y101 + 75114 Y83
   + 1200 R41 + 700 R11 + 900 R31 + 1500 R63 <=  500000
    5)  2.5 X3101 - 8 R31 <=  0
    6)  1.35 X471 + 1.03 X491 - 8 R41 <=  0
    7)  2.42 X151 + 2.48 X191 - 8 R11 <=  0
    8)  2.26 X683 - 8 R63 <=  0
    9)  Y51 + Z51 <=  1
   10)  Y71 + Z71 <=  1
   11)  Y91 + Z91 <=  1
   12)  Y101 + Z101 <=  1
   13)  Y83 + Z83 <=  1
   14)  X151 - Z51 =  0
   15)  X471 - Z71 =  0
   16)  X491 + X191 - 2 Z91 =  0
   17)  X3101 - Z101 =  0
   18)  X683 - Z83 =  0
END
INTE   X151
INTE   X471
INTE   X491
INTE   X191
INTE   X3101
INTE   X683
INTE   Y51
INTE   Y71
INTE   Y91
INTE   Y101
INTE   Y83
INTE   Z51
INTE   Z71
INTE   Z91
INTE   Z101
INTE   Z83
GIN    R41
GIN    R11
GIN    R31
GIN    R63
```

APPENDIX A.3

Pattern 3: Mathematical Formulation for the Example Problem in SuperLINDO

```
MAX    4160 X151 + 2600 X471 + 7020 X491 + 2340 X191 + 5200 X3101
    + 1820 X683 + 633.1 Y51 - 15730 Y71 + 8057.4 Y91 + 6247.8 Y101
    + 834.6 Y83 - 1200 R41 - 700 R11 - 900 R31 - 1500 R63
 SUBJECT TO
       2)  X3101 + R41 + R11 <=  5
       3)  R63 <=  6
       4)  64057.5 Y51 + 41496 Y71 + 69121 Y91 + 69641 Y101 + 75114 Y83
     + 1200 R41 + 700 R11 + 900 R31 + 1500 R63 <=  500000
       5)  2.5 X3101 - 8 R31 <=  0
       6)  1.35 X471 + 1.03 X491 - 8 R41 <=  0
       7)  2.42 X151 + 2.48 X191 - 8 R11 <=  0
       8)  2.26 X683 - 8 R63 <=  0
       9)  Y51 + Z51 <=  1
      10)  Y71 + Z71 <=  1
      11)  Y91 + Z91 <=  1
      12)  Y101 + Z101 <=  1
      13)  Y83 + Z83 <=  1
      14)  X151 - Z51 =  0
      15)  X471 - Z71 =  0
      16)  X491 + X191 - 2 Z91 =  0
      17)  X3101 - Z101 =  0
      18)  X683 - Z83 =  0
END
INTE    X151
INTE    X471
INTE    X491
INTE    X191
INTE   X3101
INTE    X683
INTE     Y51
INTE     Y71
INTE     Y91
INTE    Y101
INTE     Y83
INTE     Z51
INTE     Z71
INTE     Z91
INTE    Z101
INTE     Z83
GIN     R41
GIN     R11
GIN     R31
GIN     R63
```

APPENDIX B.

Table B.1 Machine-part load matrix for 3C*5P*4M problem structure (14 parts and 7 machines originally)

	P1	P4	P5	P6	P7	P9	P10	P3	P11	P12	P13	P2	P8	P14	Total Workload on Machine (hrs)	# of Machines
M2	0.5	0.61	0.9	2.09	1.35										5.45	1
M6	0.5			4.55									2.26		7.31	1
M7	0.55	4.74	3.61	1.47		3.87	4.68								18.92	3
M3							2.5		3.03	0.71	1.61				7.85	1
M4					1.35	1.03		3.1	0.58	0.99					7.05	1
M1			2.42			2.48						0.69	2.44	2.72	10.75	2
M5												1.22	4.45	3.84	9.51	2

Table B.2 Generated data for 3C*5P*4M problem structure (14 parts and 7 machines originally)

		prob1	prob2	prob3	prob4	prob5	prob6	prob7	prob8	prob9	prob10	prob11	prob12
Amortize cost of m/c	m1	3851	5756	5665	5881	5508	5577	3053	4786	3659	5933	4071	5788
	m2	4167	5571	4134	5166	5698	4591	3941	5696	5530	4913	4374	3679
	m3	3119	4705	3573	3886	5393	4779	4664	3962	4767	3258	3853	5378
	m4	3842	4524	3294	4554	5609	5330	3357	3774	3626	5398	5940	4146
Daily Demand	p1	737	388	569	366	615	772	540	492	370	761	473	324
	p2	727	458	571	683	727	717	796	480	560	349	463	642
	p3	456	322	572	305	763	543	626	602	451	452	423	677
	p4	750	724	640	594	425	338	368	495	495	684	635	741
	p5	418	718	534	597	736	402	318	657	766	468	444	717
Subcontracting Cost	b1	0.33	0.3	0.48	0.34	0.31	0.56	0.39	0.52	0.48	0.58	0.46	0.5
	b2	0.3	0.25	0.43	0.35	0.54	0.29	0.52	0.29	0.59	0.59	0.2	0.59
	b3	0.36	0.29	0.32	0.29	0.32	0.37	0.57	0.49	0.2	0.24	0.55	0.44
	b4	0.45	0.35	0.55	0.28	0.39	0.29	0.46	0.2	0.24	0.34	0.5	0.46
	b5	0.56	0.54	0.42	0.53	0.59	0.3	0.57	0.27	0.23	0.33	0.59	0.39
Machine Operating Cost	r1	51	50	31	37	55	30	58	46	36	58	39	38
	r2	47	41	45	54	47	34	53	41	37	58	58	54
	r3	60	51	37	48	44	33	53	48	39	35	31	44
	r4	35	53	41	52	56	35	49	44	41	46	57	53
	r5	43	44	41	54	43	36	58	30	40	34	48	36
	r6	48	34	48	39	35	35	35	53	37	49	33	59
	r7	38	43	40	50	49	39	51	36	32	40	49	58

Table B.3 Machine-part load matrix for 5C*13P*8M problem structure (42 parts and 16 machines originally)

	P2	P4	P7	P10	P18	P28	P32	P37	P38	P40	P42	P1	P5	P6	P8	P9	P11	P12	P14	P15	P16
M2	1.8			2.4		1.59	1.54	0.6	1.3	1.61	1.29										
M9	1.23	0.77		2.15	1.08	1.61	2.06	2.41	2.28	1.12	1.88										
M16	1.69		1.14	1.68	1.75		2.42	0.86	0.52		2.27										
M1								1.42			1.46										
M5													1.35		2.45	1.09			1.48	1.97	1.04
M15													1.44						1.55		
M4													1.36			0.88			1.38		
M6	1.86						1.51	1.77		1.83		0.74		1.18	1.55			0.55	1.96		
M8	0.7							2.1				1.92			1.76	2.11	0.83	1.06		2.36	
M3																					
M14	0.56																				
M7												0.97									
M10												0.75									
M11																					
M12																	2.19				
M13																					

	P19	P20	P21	P23	P29	P31	P33	P34	P39	P41	P3	P17	P35	P36	P13	P25	P26	P31	P22	P24	P27
M2																					
M9																					
M16																					
M1																					
M5	1.26		0.57	1.97	1.4		0.65			1.88	0.87										
M15	0.95		2.39				1.45			0.55											
M4	1.81		1.56	1.43	1.06																
M6	2.12			0.55			1.63	1.02	1.29						1.65						
M8	1.95	0.66	1.64	1.12		2.03				1.72								1.16			0.61
M3								1.32			0.88	0.74	1.93	2.26							
M14											1.2	2.04	1.2								
M7															2.4	2.24					
M10						0.69									0.99	1.39	2.4				
M11		0.58																1.61		1.23	1.69
M12																			0.59	0.83	0.93
M13																		2.39		2.19	

Table B.4 Generated data for 5C*13P*8M problem structure (42 parts and 16 machines originally)

		prob1	prob2	prob3	prob4	prob5	prob6	prob7	prob8	prob9	prob10	prob11	prob12
Amortize cost of m/c	m1	3918	5348	5503	3868	3499	5612	3472	4239	5656	4348	3891	4590
	m2	4023	5510	3805	4373	4588	3676	5489	4054	3303	4889	3273	5152
	m3	5433	4015	3985	3041	5671	4699	3669	3914	3711	4879	4010	3220
	m4	4689	5782	3848	4569	5187	4191	5519	3562	5932	4742	4579	5497
	m5	4941	5090	3264	3952	5457	5131	5660	5681	5033	5692	3035	5446
	m6	3975	3604	3978	4400	3481	5193	4354	3157	5131	3034	4396	5222
	m7	5973	3714	5289	5242	5406	5362	3594	5827	4048	4218	3499	4236
	m8	5387	3105	3261	3021	4044	5170	5824	5071	3905	3982	5998	5578
Daily Demand	p1	697	320	411	569	329	698	514	421	446	526	698	725
	p2	643	678	532	461	697	669	746	485	463	742	310	472
	p3	509	605	633	360	511	593	723	676	640	727	432	517
	p4	569	756	519	437	496	491	591	528	334	346	473	726
	p5	425	672	424	540	409	680	714	695	449	780	755	381
	p6	334	736	600	318	431	368	610	499	791	639	535	514
	p7	527	761	531	343	358	621	795	480	762	692	545	318
	p8	593	324	406	397	699	796	378	609	650	443	523	712
	p9	722	330	796	585	725	380	709	553	497	447	711	630
	p10	508	541	649	536	498	444	641	389	420	608	703	649
	p11	343	303	779	662	777	696	797	636	647	388	301	653
	p12	413	702	482	760	756	617	512	681	447	421	529	729
	p13	606	553	339	328	328	396	760	601	653	625	516	562
Subcontracting Cost	b1	0.23	0.32	0.55	0.29	0.37	0.43	0.42	0.28	0.32	0.45	0.57	0.5
	b2	0.24	0.48	0.46	0.56	0.34	0.56	0.5	0.25	0.35	0.53	0.26	0.59
	b3	0.41	0.29	0.45	0.54	0.27	0.58	0.3	0.38	0.28	0.39	0.42	0.51
	b4	0.34	0.43	0.52	0.21	0.26	0.29	0.53	0.42	0.33	0.48	0.57	0.55
	b5	0.59	0.47	0.49	0.25	0.52	0.59	0.34	0.2	0.29	0.2	0.52	0.28
	b6	0.53	0.27	0.32	0.39	0.58	0.39	0.59	0.52	0.38	0.22	0.55	0.42
	b7	0.45	0.59	0.36	0.5	0.48	0.58	0.53	0.53	0.25	0.42	0.41	0.38
	b8	0.46	0.51	0.47	0.23	0.3	0.3	0.34	0.55	0.58	0.46	0.54	0.49
	b9	0.56	0.38	0.3	0.28	0.34	0.32	0.58	0.43	0.36	0.35	0.29	0.25
	b10	0.41	0.3	0.51	0.57	0.34	0.24	0.51	0.29	0.24	0.46	0.25	0.24
	b11	0.55	0.3	0.41	0.3	0.31	0.32	0.25	0.38	0.3	0.53	0.33	0.57
	b12	0.48	0.32	0.43	0.41	0.51	0.31	0.27	0.49	0.22	0.53	0.48	0.37
	b13	0.37	0.45	0.45	0.43	0.27	0.49	0.55	0.51	0.3	0.55	0.41	0.54
Machine Operating Cost	r1	58	44	58	43	55	39	43	55	38	32	55	42
	r2	35	47	30	44	42	42	48	31	36	41	34	37
	r3	49	46	48	57	53	34	39	42	49	52	43	46
	r4	55	31	32	31	36	49	36	43	48	51	40	48
	r5	58	46	33	47	31	48	51	35	44	48	54	48
	r6	48	47	44	36	60	38	58	54	34	47	43	45
	r7	40	49	43	41	31	60	33	48	47	34	38	50
	r8	31	33	30	46	56	50	46	55	36	39	32	37
	r9	40	42	54	45	51	30	30	48	44	41	40	31
	r10	47	41	32	41	40	38	43	51	44	39	58	54
	r11	39	32	58	33	59	36	35	55	53	47	60	52
	r12	43	43	58	33	56	53	56	36	58	40	42	36
	r13	48	49	43	37	47	45	60	36	39	48	52	41
	r14	45	43	40	34	52	49	46	49	35	56	47	46
	r15	37	55	32	31	33	46	53	43	59	38	32	51
	r16	44	40	50	56	55	32	34	33	31	33	45	59

Table B.5 Machine-part load matrix for 8C*27P*21M problem structure (80 parts and 40 machines originally)

	P4	P5	P9	P24	P33	P39	P49	P57	P58	P65	P66	P12	P13	P54	P61	P64	P73	P77	P78	P3	P10	P19	P20	P36	P48	P50	P6	P17
M1	1.67	0.87	0.94	0.65	0.66		1.03	1.44	1.66	1.63	1.74																	
M3	0.62	1.1		1.96	0.65	1.38	1.81	1.16	0.79	1.66	1.17																	
M7	0.56	0.82	0.66	1.38	1.68	0.66	1.34	0.61		1.26	0.78																	
M32	1.23	1.25	0.92	1.47		1.91	0.81	0.65	1.55	1.64	1																	
M2												0.98	1.4	1.78	1.12	1.58		0.59	0.56									
M10												1.86	0.86	0.64	0.83	1.68	0.69	1.04	0.58									
M16												1.91	1.13	0.69	1.58		0.63	0.87	1.09									
M21												1.14	0.78		1.11	1.25	1.28	1.54	0.77									
M31												1.47		1.34	1.34	1.14	1.31	1.37										
M4																				1.7	1.65	1.78	1.58	0.86	1.07	0.53		
M9																				0.54	1.87	0.51	0.84	0.57	1.16	1.85		
M20																				1.77	1.18	1.07		0.77	1.2	0.57		
M5																											1.31	1.34
M8																											1.03	1.42
M22																											0.98	0.65
M23																	1.96										0.79	0.64
M37																											1.56	0.9
M39																											1.69	1.5
M6																												
M12								1.25																				
M26															1.06													
M38																												
M40																			1.07									
M11																												
M13																												
M14																												
M17																												
M35																												
M15																												
M18																												
M33																												
M34																												
M36																	1.51	1.93										
M19																												
M25																												
M28																												
M30																						0.94						
M24																												
M27						1.45																						
M29																												

Table B.5 Machine-part load matrix for 8C*27P*21M problem structure (80 parts and 40 machines originally) (continued)

P26	P27	P28	P46	P56	P69	P70	P76	P1	P2	P14	P15	P29	P30	P38	P40	P43	P44	P45	P59	P60	P62	P63	P7	P11	P18	P37	P42	P56
																										1.62		0.96
											1.85								1.45									
														1.19														
			0.66																									
1.25	0.87	1.94	1.52	1.02	1.47	1.72	1.26													1.69								
0.99	1.94		1.22	1.75	0.87	0.84	0.88												0.72									
0.62	1.59	0.79	1.42	0.74	1.17	1.25	1.86																					
1.86	1.43	1.03	0.83	1.94	0.65	0.84	1.02																					
1.54	1.29	1.24	1.32	0.6	0.98	1.45	0.8																					
0.88		0.84		1.73	1.38	0.62	1.09																					
								1.91	0.53		1.08	1.11	0.91	0.6	1.44	1.3	0.69	1.6	1.04	1.66	0.88							
								0.91	0.94	1.02		0.91	1.44	0.92		0.54	1.74	1.46	0.72	1.43	1.32	0.76						
								1.61		0.71	1.22	0.62	0.61	1.36	0.52	0.67	1.31	1.43	0.96	1.9	0.94	0.65						
								1.75	0.57	0.73	1.14	0.56	1.04		1.96	1.61		1.58	0.71	0.63	1.87	0.87						
								1.99	0.54	0.55	1.63		0.76	1.01	1.71	1.81	1.82	1.21		1.29		1.5						
																							1.11	0.79	1.47	0.88	1.75	0.84
																							0.97	0.64	0.51	1.96	1.6	1.97
																							1.84		1.43	0.94		
									1.3					1.64			1.92						0.72		1.1	1.69	1.65	1.48
																							1.56	1.91	1.67		0.65	1.95
																												0.57
							1.91							1.04			0.61											
1.62																												
									1.71																			

Table B.5 Machine-part load matrix for 8C*27P*21M problem structure (80 parts and 40 machines originally) (continued)

P67	P79	P80	P21	P22	P52	P75	P23	P31	P32	P41	P51	P74	P71	P72	P8	P16	P25	P34	P35	P47	P53	P68
1.66																						
						0.94												1.59				
1.06	1.28	1.33	1.69	1.65		1.53														1.32		0.66
1.16	0.66	1.67	0.61		0.77	0.71																
		1.67	0.75	1.97	1.29	1.31			0.63													
1.44	0.6	1.27	1.35	0.85	0.96																	
	1.33		0.52	0.8	0.73	0.52																
1.08							1.39	0.8			1.13	1.98										
							1.37	1.58	1.66	0.93		1.46										
							0.96	1.48		1.86	1.31	1.55										
							1.99	1.7	0.59	0.77	1.75	0.8										
							1.4		1.29	1.99	1.84	0.74										
									0.74	1.66			1.53	1.88	1.23	1.5		0.8	0.67			
								1.75					0.63	0.53	1.15		1.38	1.86	1.45			1.58
													0.65	1.99	1.17			0.62	0.52		1.34	
												1.56	0.82	1.76		1.18	1.03	1.03		1.23		0.82
														1.16	1.69	1.31	1.98	1.5	0.71	0.75	1.06	1.48
													1.99		1.17	1.83	1.85	1.79	0.6	1.98	1.22	0.75
													0.55		1.36	0.57	1.59	0.57	0.6	1.17	1.82	1.4

Table B.6 Generated data for 8C*27P*21M problem structure (80 parts and 40 machines originally)

		prob1	prob2	prob3	prob4	prob5	prob6	prob7	prob8	prob9	prob10	prob11	prob12
Amortize cost of m/c	m1	5714	5949	3789	4000	5858	4735	4399	5502	5851	5086	3664	3603
	m2	3207	4091	3582	3575	4661	5456	3373	4941	5854	3384	3068	5595
	m3	4701	3025	3421	3090	3481	5667	3896	4715	3668	3143	3100	5626
	m4	3006	5212	4557	4179	3305	5045	5379	4228	5517	4097	5389	5230
	m5	5144	4461	5658	5461	5568	4329	5167	5604	3543	4905	3549	5516
	m6	3491	4348	3393	5719	5328	3964	4863	5204	3157	5474	4878	4369
	m7	5251	4416	3789	4785	4671	5118	4510	5235	5737	3974	4969	3892
	m8	3793	3025	4629	5049	4921	5021	4888	4182	3481	4330	4775	3362
	m9	5702	3858	5506	4031	4404	4496	3232	4239	4476	5407	4685	5723
	m10	3993	3787	3115	3777	5216	4657	3015	3857	5137	3198	4023	4366
	m11	4679	4900	4806	5586	5413	3133	4498	4986	5823	3425	3606	3172
	m12	5823	5671	5836	3890	3570	5032	3908	5932	3583	3903	4436	5143
	m13	3496	3349	4081	5410	4838	4423	4711	5138	5149	4724	5838	5352
	m14	5975	5864	3779	3804	4416	4192	4544	5993	3839	3742	3505	3008
	m15	4608	4703	5937	4347	3991	3515	5663	4261	5402	5888	3880	4724
	m16	4253	3918	5735	4053	5751	3938	4977	3434	4322	5524	5564	4184
	m17	5979	3761	5018	5157	3086	3182	3283	5285	5264	5040	5631	5255
	m18	5159	5254	5885	5706	5113	3398	5337	3447	4188	5399	5588	4623
	m19	4479	5091	3272	3148	3497	5811	5391	4625	4837	5126	3371	4624
	m20	5632	3748	4744	3448	3011	4321	4323	4198	4893	5509	5956	4157
	m21	3938	3965	3246	4099	5317	5239	5818	5387	3475	4387	3194	4100
Daily Demand	p1	367	711	328	552	408	480	455	422	590	559	737	394
	p2	741	334	639	491	705	763	570	662	788	789	429	412
	p3	412	333	366	704	781	670	314	697	399	778	343	339
	p4	540	598	305	357	640	416	608	506	319	631	520	731
	p5	746	673	515	716	742	308	645	580	588	748	698	510
	p6	795	391	591	646	742	510	595	318	599	786	424	353
	p7	763	501	783	691	492	357	591	324	451	492	772	657
	p8	334	665	497	341	486	653	490	544	460	473	611	626
	p9	445	565	546	447	566	716	583	671	730	704	655	740
	p10	446	692	373	753	331	463	409	744	326	676	374	547
	p11	538	617	396	467	487	580	696	562	388	577	615	655
	p12	424	493	332	477	314	431	539	314	603	398	720	339
	p13	534	680	435	388	417	788	551	309	728	515	437	700
	p14	658	776	307	592	624	690	351	537	782	653	525	787
	p15	691	421	390	442	689	535	426	348	781	743	561	478
	p16	584	541	762	527	449	527	376	761	737	469	535	651
	p17	748	356	461	386	461	695	337	339	626	655	556	753
	p18	681	777	791	692	322	469	560	723	619	330	345	714
	p19	534	725	409	643	575	421	581	754	398	558	687	378
	p20	537	337	508	404	672	749	391	457	493	363	609	677
	p21	529	603	393	461	701	442	366	306	781	351	353	744
	p22	732	439	765	380	471	444	706	362	531	749	777	379
	p23	791	432	313	346	480	332	483	516	777	582	771	691
	p24	307	601	556	449	517	480	458	429	764	763	398	448
	p25	486	447	482	731	606	488	337	307	372	339	624	459
	p26	594	665	762	721	628	535	798	722	523	669	469	301
	p27	679	343	499	734	636	331	421	507	739	558	425	677
Subcontracting Cost	b1	0.6	0.57	0.55	0.31	0.55	0.49	0.4	0.23	0.5	0.27	0.59	0.49
	b2	0.22	0.38	0.51	0.21	0.24	0.47	0.49	0.42	0.29	0.47	0.34	0.45
	b3	0.59	0.41	0.24	0.51	0.4	0.33	0.6	0.4	0.36	0.42	0.23	0.26
	b4	0.33	0.48	0.47	0.24	0.34	0.52	0.29	0.51	0.39	0.51	0.44	0.29
	b5	0.39	0.39	0.45	0.43	0.55	0.45	0.58	0.36	0.24	0.24	0.29	0.41
	b6	0.56	0.33	0.46	0.25	0.35	0.54	0.45	0.47	0.52	0.39	0.2	0.25
	b7	0.43	0.22	0.22	0.45	0.49	0.24	0.33	0.45	0.53	0.34	0.27	0.33
	b8	0.56	0.46	0.3	0.56	0.45	0.21	0.32	0.41	0.35	0.28	0.3	0.54
	b9	0.24	0.56	0.21	0.44	0.49	0.45	0.38	0.24	0.51	0.22	0.5	0.5
	b10	0.5	0.5	0.56	0.24	0.22	0.56	0.43	0.54	0.25	0.39	0.46	0.51
	b11	0.59	0.32	0.36	0.2	0.2	0.29	0.4	0.47	0.34	0.5	0.48	0.42
	b12	0.26	0.42	0.59	0.37	0.2	0.31	0.52	0.35	0.21	0.41	0.37	0.59
	b13	0.25	0.21	0.52	0.22	0.42	0.37	0.4	0.33	0.5	0.31	0.52	0.42
	b14	0.54	0.24	0.56	0.32	0.34	0.56	0.53	0.37	0.25	0.38	0.37	0.41
	b15	0.24	0.27	0.52	0.23	0.53	0.57	0.52	0.35	0.3	0.54	0.29	0.57
	b16	0.26	0.28	0.38	0.59	0.36	0.39	0.44	0.51	0.57	0.45	0.4	0.27
	b17	0.51	0.49	0.44	0.24	0.25	0.25	0.3	0.5	0.52	0.56	0.29	0.55
	b18	0.22	0.6	0.38	0.43	0.47	0.33	0.43	0.22	0.26	0.45	0.22	0.25
	b19	0.38	0.43	0.56	0.51	0.49	0.57	0.58	0.26	0.53	0.47	0.3	0.38
	b20	0.54	0.59	0.2	0.31	0.58	0.53	0.3	0.32	0.55	0.27	0.6	0.33
	b21	0.25	0.23	0.31	0.23	0.51	0.28	0.53	0.26	0.52	0.25	0.28	0.55
	b22	0.59	0.46	0.35	0.37	0.51	0.41	0.27	0.37	0.44	0.54	0.28	0.59
	b23	0.4	0.49	0.56	0.21	0.24	0.27	0.55	0.44	0.33	0.46	0.52	0.2
	b24	0.35	0.53	0.26	0.57	0.44	0.32	0.31	0.44	0.24	0.28	0.38	0.37
	b25	0.4	0.55	0.3	0.57	0.36	0.28	0.55	0.39	0.49	0.37	0.28	0.29
	b26	0.23	0.28	0.21	0.53	0.37	0.26	0.48	0.28	0.5	0.39	0.55	0.26
	b27	0.33	0.32	0.58	0.33	0.42	0.53	0.44	0.6	0.41	0.32	0.5	0.39

Table B.6 Generated data for 8C*27P*21M problem structure (80 parts and 40 machines originally) (continued)

Machine Operating Cost	r1	35	31	44	55	37	53	44	53	42	31	54	40
	r2	49	50	41	43	35	38	35	31	40	31	34	45
	r3	50	56	50	56	32	40	37	38	59	42	42	46
	r4	54	32	33	39	43	56	53	55	39	50	54	37
	r5	51	51	45	34	36	51	52	48	49	52	57	40
	r6	35	42	47	50	47	53	58	42	39	51	52	44
	r7	53	40	46	36	53	39	42	31	57	38	33	55
	r8	44	44	50	46	50	45	34	52	41	37	30	53
	r9	34	50	37	31	54	49	31	35	36	47	36	30
	r10	56	54	47	52	40	38	44	50	58	45	34	40
	r11	44	59	50	34	58	55	45	58	32	35	34	31
	r12	36	40	53	54	38	57	34	54	48	47	43	56
	r13	51	60	46	53	60	59	52	41	46	57	31	41
	r14	46	37	48	46	36	53	49	33	34	39	48	52
	r15	40	54	40	60	50	58	36	47	41	36	43	57
	r16	37	34	46	45	45	47	43	56	48	57	51	34
	r17	42	49	55	36	36	36	34	31	44	55	46	47
	r18	50	31	36	34	33	58	46	38	30	55	39	35
	r19	33	57	32	44	59	57	34	49	56	52	59	42
	r20	35	48	35	37	40	54	53	37	53	48	55	36
	r21	49	51	35	52	31	35	35	48	39	42	45	39
	r22	51	53	59	46	53	55	58	56	31	51	56	56
	r23	60	39	38	58	46	40	34	42	44	43	35	35
	r24	57	48	33	53	43	55	47	47	34	36	47	35
	r25	43	32	39	42	46	37	40	59	55	43	59	36
	r26	50	53	36	34	42	33	59	31	43	35	44	37
	r27	32	47	57	53	51	31	32	38	41	45	43	32
	r28	56	33	52	53	39	50	45	45	34	35	57	47
	r29	38	44	54	44	41	57	33	51	38	42	55	37
	r30	42	47	44	58	41	30	48	38	54	59	39	50
	r31	51	45	48	40	45	47	51	38	40	42	53	44
	r32	40	31	52	44	34	58	55	30	38	49	51	52
	r33	48	56	56	35	52	40	52	42	40	41	35	31
	r34	60	45	56	39	49	42	37	41	59	44	44	49
	r35	56	51	37	34	43	48	42	48	46	47	57	52
	r36	40	54	35	34	50	51	42	35	58	47	58	42
	r37	32	38	45	46	34	59	41	54	37	43	41	44
	r38	38	35	32	39	40	46	48	52	52	47	44	39
	r39	34	48	55	56	55	49	46	36	49	31	50	38
	r40	51	37	59	54	50	46	55	56	38	30	38	57

Table B.7 Parameters used in tabu search-based heuristics for each problem structure (single row and double row layout)

Parameters	3C*5P*4M		5C*13P*8M		8C*27P*21M	
	Inside search	Outside search	Inside search	Outside search	Inside search	Outside search
Tabu List Size	Fixed : 3 Variable : -initial : 3 -decreased : 2 -increased : 4	Fixed : 1 No Variable :	Fixed : 8 Variable : -initial : 8 -decreased : 6 -increased : 10	Fixed : 2 Variable : -initial : 2 -decreased : 1 -increased : 3	Fixed : 16 Variable : -initial : 16 -decreased : 12 -increased : 20	Fixed : 4 Variable : -initial : 4 -decreased : 2 -increased : 5
Number of Iterations w/o Improvement	2	6	6	12	12	15
Number of Restarts	2	2	2	2	2	2

APPENDIX C.

Table C.1 Results obtained for 3C*5P*4M problem structure (Single-row layout)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	145536	145536	145536	145536	145536	145536
Problem 2	150077	150077	150077	150077	150077	150077
Problem 3	53957.8	53957.8	53957.8	53957.8	53957.8	53957.8
Problem 4	95004.2	95004.2	95004.2	95004.2	95004.2	95004.2
Problem 5	173725	173725	173725	173725	173725	173725
Problem 6	128536	128536	128536	128536	128536	128536
Problem 7	77934	77934	77934	77934	77934	77934
Problem 8	87156.5	87156.5	87156.5	87156.5	87156.5	87156.5
Problem 9	81117.6	81117.6	81117.6	81117.6	81117.6	81117.6
Problem 10	81920.8	81920.8	81920.8	81920.8	81920.8	81920.8
Problem 11	26475	26475	26475	26475	26475	26475
Problem 12	3134	3134	3134	3134	3134	3134
Avg. Total Savings	92047.83	92047.83	92047.83	92047.83	92047.83	92047.83

CPU Time (sec)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	0.1	0.3	0.3	0.1	0.3	0.3
Problem 2	0.3	0.6	0.5	0.2	0.4	0.4
Problem 3	0.2	0.6	0.6	0.3	0.7	0.6
Problem 4	0.2	0.7	0.4	0.2	0.5	0.4
Problem 5	0.1	0.3	0.2	0.1	0.3	0.3
Problem 6	0.1	0.1	0.2	0.2	0.3	0.3
Problem 7	1	1.4	1.4	0.3	0.5	0.5
Problem 8	0.1	0.3	0.3	0.2	0.3	0.3
Problem 9	0.5	1	0.7	0.2	0.4	0.5
Problem 10	0.2	0.4	0.4	0.2	0.5	0.4
Problem 11	0.2	0.9	0.4	0.2	0.4	0.4
Problem 12	0.1	0.3	0.2	0.2	0.4	0.5

Table C.2 Results obtained for 3C*5P*4M problem structure (Double-row layout)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	145536	145536	145536	145536	145536	145536
Problem 2	150077	150077	150077	150077	150077	150077
Problem 3	53957.8	53957.8	53957.8	53957.8	53957.8	53957.8
Problem 4	95004.2	95004.2	95004.2	95004.2	95004.2	95004.2
Problem 5	173725	173725	173725	173725	173725	173725
Problem 6	128536	128536	128536	128536	128536	128536
Problem 7	77934	77934	77934	77934	77934	77934
Problem 8	87156.5	87156.5	87156.5	87156.5	87156.5	87156.5
Problem 9	81117.6	81117.6	81117.6	81117.6	81117.6	81117.6
Problem 10	81920.8	81920.8	81920.8	81920.8	81920.8	81920.8
Problem 11	26475	26475	26475	26475	26475	26475
Problem 12	3134	3134	3134	3134	3134	3134
Avg. Total Savings	92047.83	92047.83	92047.83	92047.83	92047.83	92047.83

CPU Time (sec)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	0.1	0.3	0.3	0.1	0.3	0.3
Problem 2	0.3	0.6	0.5	0.2	0.4	0.4
Problem 3	0.2	0.6	0.6	0.3	0.7	0.6
Problem 4	0.2	0.7	0.4	0.2	0.5	0.4
Problem 5	0.1	0.3	0.2	0.1	0.3	0.3
Problem 6	0.1	0.1	0.2	0.2	0.3	0.3
Problem 7	1	1.4	1.4	0.3	0.5	0.5
Problem 8	0.1	0.3	0.3	0.2	0.3	0.3
Problem 9	0.5	1	0.7	0.2	0.4	0.5
Problem 10	0.2	0.4	0.4	0.2	0.5	0.4
Problem 11	0.2	0.9	0.4	0.2	0.4	0.4
Problem 12	0.1	0.3	0.2	0.2	0.4	0.5

Table C.3 Results obtained for 5C*13P*8M problem structure (Single-row layout)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	348848	348848	348848	348848	348848	348848
Problem 2	370989	370989	370989	370989	370989	370989
Problem 3	226390	230229	226390	226390	230229	226390
Problem 4	260698	260698	260698	260698	260698	260698
Problem 5	487509	487509	487509	487509	487509	487509
Problem 6	203291	203291	203291	203291	203291	203291
Problem 7	335610	335610	335610	335610	335610	335610
Problem 8	366278	366278	366278	366278	366278	366278
Problem 9	305014	305014	305014	305014	305014	305014
Problem 10	197124	197124	197124	197124	197124	197124
Problem 11	275526	275526	275526	275526	275526	275526
Problem 12	214835	214835	214835	214835	214835	214835
Avg. Total Savings	299342.7	299662.6	299342.7	299342.7	299662.6	299342.7

CPU Time (h:mm:ss)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	00:00:29	00:02:40	00:01:35	00:01:18	00:06:26	00:03:45
Problem 2	00:00:39	00:46:47	00:43:26	00:02:26	00:07:28	00:05:55
Problem 3	00:29:23	02:40:07	02:31:10	00:02:29	00:07:47	00:06:52
Problem 4	00:00:48	00:39:25	00:50:17	00:02:21	00:07:13	00:07:54
Problem 5	00:21:06	02:41:41	02:12:41	00:01:36	00:05:09	00:05:7
Problem 6	00:00:46	00:06:59	00:01:43	00:01:25	00:04:50	00:03:45
Problem 7	00:46:50	03:17:48	04:25:16	00:02:12	00:08:12	00:07:8
Problem 8	00:25:35	00:25:35	01:43:41	00:02:17	00:06:27	00:05:57
Problem 9	00:59:20	03:32:58	04:08:09	00:02:41	00:08:24	00:09:30
Problem 10	00:00:36	02:31:53	02:13:58	00:01:59	00:09:07	00:06:42
Problem 11	00:34:40	02:36:55	03:35:08	00:01:52	00:05:05	00:05:44
Problem 12	00:10:36	01:17:40	01:45:01	00:02:18	00:06:10	00:04:44

Table C.4 Results obtained for 5C*13P*8M problem structure (Double-row layout)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	272863	272863	272863	272863	272863	272863
Problem 2	290454	290454	290454	290454	290454	290454
Problem 3	180132	168349	168349	183819	163409	163409
Problem 4	194690	194690	194690	209738	191942	191942
Problem 5	431089	431089	431089	415164	411975	411979
Problem 6	142321	142321	157791	153651	135041	135041
Problem 7	253860	253860	253860	267285	253860	253860
Problem 8	321688	321688	321688	321688	301668	301668
Problem 9	237998	237998	237998	253144	235399	235399
Problem 10	155264	155264	155264	155264	134315	133886
Problem 11	229683	222403	222403	229683	222403	222403
Problem 12	140670	140670	140670	154320	140670	140670
Avg. Total Savings	237559.3	235970.8	237259.9	242256.1	229499.9	229464.5

CPU Time (h:mm:ss)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	00:00:13	00:04:05	00:00:43	00:00:22	00:06:16	00:04:22
Problem 2	00:00:20	00:37:13	00:34:11	00:00:39	00:08:35	00:06:19
Problem 3	00:21:01	01:41:24	01:44:35	00:02:07	00:11:17	00:12:43
Problem 4	00:00:30	00:14:25	00:13:22	00:01:12	00:07:30	00:06:15
Problem 5	00:11:02	01:10:08	01:08:33	00:01:14	00:07:51	00:05:05
Problem 6	00:05:05	00:01:05	00:01:03	00:00:40	00:06:26	00:05:51
Problem 7	00:12:48	00:58:38	00:58:47	00:01:26	00:11:46	00:12:03
Problem 8	00:26:32	02:41:40	02:17:40	00:02:04	00:08:20	00:08:58
Problem 9	00:38:07	02:07:30	02:21:42	00:02:05	00:20:30	00:25:31
Problem 10	00:22:35	01:22:13	01:42:35	00:02:02	00:07:43	00:09:50
Problem 11	00:23:50	01:21:44	01:38:41	00:02:11	00:14:27	00:07:10
Problem 12	00:01:06	00:18:40	00:25:04	00:01:49	00:03:31	00:02:14

Table C.5 Results obtained for 8C*27P*21M problem structure (Single-row layout)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	1841090	1841090	1841090	1841090	1841090	1841090
Problem 2	1446440	1446440	1446440	1446440	1446440	1446440
Problem 3	1284850	1284850	1284850	1284850	1284850	1284850
Problem 4	1213520	1216980	1213520	1215610	1217280	1215610
Problem 5	1801680	1801680	1801680	1801680	1801680	1801680
Problem 6	1887450	1887450	1887450	1887450	1887450	1887450
Problem 7	1319400	1319400	1319400	1318950	1318950	1318950
Problem 8	1356710	1359400	1356710	1359440	1359440	1359440
Problem 9	1313420	1319270	1330070	1295230	1322570	1334590
Problem 10	1242500	1246140	1242500	1242500	1246140	1242500
Problem 11	1147200	1147200	1147200	1147200	1149040	1147200
Problem 12	101900	101900	101900	1018850	1019000	1018850
Avg. Total Savings	1329680	1330983	1331068	1404941	1407828	1408221

CPU Time (h:mm:ss)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	1:43:47	10:18:47	8:04:16	2:02:39	15:45:06	13:12:22
Problem 2	1:58:01	13:29:32	12:11:31	1:59:36	15:18:34	14:22:52
Problem 3	2:42:37	13:11:39	11:09:27	2:12:11	13:13:20	8:13:29
Problem 4	1:46:44	10:24:52	10:51:01	1:20:54	8:16:28	10:01:19
Problem 5	1:55:11	10:39:25	7:59:30	1:58:41	14:50:28	10:22:21
Problem 6	1:55:50	6:47:12	9:50:40	1:37:23	14:18:38	14:54:42
Problem 7	1:52:28	9:54:36	10:17:53	2:24:24	9:42:42	16:15:43
Problem 8	2:04:18	13:16:56	13:07:20	2:08:55	12:12:23	13:19:00
Problem 9	1:44:17	10:54:51	13:54:12	1:45:32	10:48:35	15:23:35
Problem 10	1:47:04	11:05:45	11:40:27	1:53:27	11:30:56	12:14:18
Problem 11	1:35:00	13:09:55	9:17:29	2:09:10	11:33:09	11:09:25
Problem 12	2:09:44	11:43:22	10:18:16	1:33:05	13:17:42	11:59:21

Table C.6 Results obtained for 8C*27P*21M problem structure (Double-row layout)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	1435230	1435230	1435230	1466110	1305410	1309410
Problem 2	986720	1039820	1001370	988407	911592	910717
Problem 3	925766	937772	925766	925766	824687	815958
Problem 4	845843	867843	845843	866812	760540	755151
Problem 5	1406280	1406280	1406280	1357600	1273860	1273860
Problem 6	1477080	1508020	1491180	1477080	1384640	1380650
Problem 7	964049	966324	964049	917428	855921	831910
Problem 8	992869	1021830	996716	992869	909507	891799
Problem 9	924527	940431	924284	892768	834855	831368
Problem 10	928893	932645	928893	929907	835679	826406
Problem 11	793206	766577	793206	766001	689710	687855
Problem 12	641654	677929	648024	638113	594065	588048
Avg. Total Savings	1026843	1041725	1030070	1018238	931705.5	925261

CPU Time (h:mm:ss)

	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	1:19:30	10:06:29	8:12:39	1:13:40	10:30:24	9:47:22
Problem 2	1:39:29	7:56:32	10:16:35	1:23:35	11:03:38	9:35:29
Problem 3	1:46:38	11:45:51	9:31:58	1:49:57	7:02:06	8:24:39
Problem 4	1:11:43	8:38:09	6:10:55	1:14:46	5:37:22	7:09:38
Problem 5	1:04:37	9:57:47	10:06:26	1:16:07	11:43:13	10:53:06
Problem 6	1:00:05	7:52:21	7:46:33	1:12:31	6:29:02	8:10:50
Problem 7	1:05:46	9:25:17	7:19:44	1:20:58	10:22:06	8:15:59
Problem 8	1:25:24	6:37:09	7:53:18	1:22:58	10:47:26	8:32:06
Problem 9	1:12:43	4:14:42	7:21:39	0:56:51	8:04:25	7:43:07
Problem 10	1:14:05	8:44:18	7:27:11	1:17:21	8:53:42	7:17:17
Problem 11	1:30:42	7:16:39	5:37:24	1:10:36	6:51:13	6:35:59
Problem 12	1:15:01	7:11:18	6:47:37	1:13:34	7:10:40	7:59:35

APPENDIX D.

Table D.1 Results obtained from analysis of variance for 5C*13P*8M problem structure (single-row layout)

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-ratio
MAIN EFFECTS - Treatments (TSH)	1.63755e+06	5	327509	1
- Blocks (Problems)	4.96992e+11	11	4.51811e+10	137954
RESIDUAL (Error)	1.8013e+07	55	327509	
TOTAL (CORRECTED)	4.97012e+11	71		

Contrast	Differences	LSD limits
TSH 1 - TSH 2	-319.917	468.214
TSH 1 - TSH 3	0	468.214
TSH 1 - TSH 4	0	468.214
TSH 1 - TSH 5	-319.917	468.214
TSH 1 - TSH 6	0	468.214
TSH 2 - TSH 3	319.917	468.214
TSH 2 - TSH 4	319.917	468.214
TSH 2 - TSH 5	0	468.214
TSH 2 - TSH 6	319.917	468.214
TSH 3 - TSH 4	0	468.214
TSH 3 - TSH 5	-319.917	468.214
TSH 3 - TSH 6	0	468.214
TSH 4 - TSH 5	-319.917	468.214
TSH 4 - TSH 6	0	468.214
TSH 5 - TSH 6	319.917	468.214

Table D.2 Results obtained from analysis of variance for 5C*13P*8M problem structure (double-row layout)

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-ratio
MAIN EFFECTS - Treatments (TSH)	1.50559e+09	5	3.01117e+08	9.32654
- Blocks (Problems)	4.57555e+11	11	4.115959e+10	1288.35
RESIDUAL (Error)	1.77573e+09	55	3.22861e+07	
TOTAL (CORRECTED)	4.60837e+11	71		

Contrast	Differences	LSD limits
TSH 1 - TSH 2	1588.58	4648.79
TSH 1 - TSH 3	299.417	4648.79
TSH 1 - TSH 4	-4696.75	4648.79
TSH 1 - TSH 5	8059.08	4648.79
TSH 1 - TSH 6	8094.83	4648.79
TSH 2 - TSH 3	-1289.17	4648.79
TSH 2 - TSH 4	-6285.33	4648.79
TSH 2 - TSH 5	6470.5	4648.79
TSH 2 - TSH 6	6506.25	4648.79
TSH 3 - TSH 4	-4996.17	4648.79
TSH 3 - TSH 5	7759.67	4648.79
TSH 3 - TSH 6	7795.42	4648.79
TSH 4 - TSH 5	12755.8	4648.79
TSH 4 - TSH 6	12791.6	4648.79
TSH 5 - TSH 6	35.75	4648.79

Table D.3 Results obtained from analysis of variance for 8C*27P*21M problem structure (single-row layout)

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-ratio
MAIN EFFECTS - Treatments (TSH)	9.22896e+07	5	1.84579e+07	1.09586
- Blocks (Problems)	5.34201e+12	11	4.85637e+11	28832.6
RESIDUAL (Error)	9.26384e+08	55	1.63433e+07	
TOTAL (CORRECTED)	5.34303e+12	71		

Contrast	Differences	LSD limits
TSH 1 - TSH 2	-1306.67	3357.74
TSH 1 - TSH 3	-1387.5	3357.74
TSH 1 - TSH 4	1189.17	3357.74
TSH 1 - TSH 5	-1697.5	3357.74
TSH 1 - TSH 6	-2115.83	3357.74
TSH 2 - TSH 3	-80.8333	3357.74
TSH 2 - TSH 4	2495.83	3357.74
TSH 2 - TSH 5	-390.833	3357.74
TSH 2 - TSH 6	-809.167	3357.74
TSH 3 - TSH 4	2576.67	3357.74
TSH 3 - TSH 5	-310	3357.74
TSH 3 - TSH 6	-728.333	3357.74
TSH 4 - TSH 5	-2886.67	3357.74
TSH 4 - TSH 6	-3305	3357.74
TSH 5 - TSH 6	-418.333	3357.74

Table D.4 Results obtained from analysis of variance for 8C*27P*21M problem structure (double-row layout)

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-ratio
MAIN EFFECTS - Treatments (TSH)	1.6495e+11	5	3.299e+10	2.80889
- Blocks (Problems)	4.73044e+12	11	4.3004e+11	36.6151
RESIDUAL (Error)	6.45967e+11	55	1.17449e+10	
TOTAL (CORRECTED)	5.54135e+12	71		

Contrast	Differences	LSD limits
TSH 1 - TSH 2	-88886	88665.9
TSH 1 - TSH 3	-77231	88665.9
TSH 1 - TSH 4	-65400.2	88665.9
TSH 1 - TSH 5	20801	88665.9
TSH 1 - TSH 6	27578.1	88665.9
TSH 2 - TSH 3	11655	88665.9
TSH 2 - TSH 4	23485.8	88665.9
TSH 2 - TSH 5	109687	88665.9
TSH 2 - TSH 6	116464	88665.9
TSH 3 - TSH 4	11830.8	88665.9
TSH 3 - TSH 5	98032	88665.9
TSH 3 - TSH 6	104809	88665.9
TSH 4 - TSH 5	86201.2	88665.9
TSH 4 - TSH 6	92978.3	88665.9
TSH 5 - TSH 6	6777.08	88665.9

APPENDIX E.

APPENDIX E.1: PSEUDO CODE FOR TABU SEARCH-BASED HEURISTIC ALGORITHM.

```
OUTSIDE SEARCH
Generate the initial cell locations configuration
Evaluate the total saving for the initial cell location configuration by Call subroutine (INSIDE SEARCH)
Initialize the outside candidate list (OCL) and the outside index list (OIL)
Initialize the outside long-term memory (OUT_LTM frequency matrix)    // all heuristics except TSH 1 and TSH 4//
do
{
        Initialize the outside tabu-list (out_tabu list)
        do
        {
                Evaluate the cell locations seeds configuration
                Evaluate the total saving for each cell locations seed configuration by Call subroutine (INSIDE
                SEARCH)
                Use the evaluated total saving to sort the seeds of cell location configuration in a non-decreasing
                order
                for ( all sorted cell location configuration seeds )
                {
                        the best solution outside ← 0
                        check against OCL
                        if (out_move status ≠ tabu) or (out_move status = tabu but out_AL criteria is satisfied)
                        {
                                out_tabu list ← location of cell that was moved to the next adjacent position
                                OCL ← the current cell locations configuration
                                update out_AL
                        }
                }
                update OIL
                update OUT_LTM frequency matrix    // all heuristics except TSH 1 and TSH 4 //
                if (there is an improvement in total saving)
                {
                        out_iter_w/o_improvement = 0
                        update best solution outside
                }
                Else
                {
                        out_iter_w/o_improvement = out_iter_w/o_improvement + 1
                }
        } while (out_iter_w/o_improvement < OIT)
        Identify the new restart using OUT_LTM frequency matrix    // all heuristics except TSH 1 and TSH 4 //
} while ( the number of restart < 2)
```

APPENDIX E.1: PSEUDO CODE FOR TABU SEARCH-BASED HEURISTIC ALGORITHM (CONTINUED).

```
subroutine ( INSIDE SEARCH)
Generate the initial part options configuration by selecting the maximum contributing option for each bottleneck part
Evaluate the total saving for the initial part options configuration
Initialize the inside candidate list (ICL) and inside index list (IIL)
Initialize the inside long-term memory (IN_LTM frequency matrix)        // all heuristics except TSH 1 and TSH 4 //
do
{
        Initialize the inside tabu-list (in_tabu list)
        do
        {
                Evaluate the part options seed configurations
                Evaluate the total saving of each part options seed configuration
                Use the evaluated total saving to sort the seeds of part options configuration in a non-decreasing
                order
                for (all sorted part option configuration seeds)
                {
                        the best solution inside ← 0
                        Check against ICL
                        If (in_move status ≠ tabu) or (in_move status = tabu but in_AL criteria is satisfied)
                        {
                                in_tabu list ← part and option that was moved
                                ICL ← the current part option configuration
                                update the in_AL
                        }
                Update IIL
                Update IN_LTM frequency matrix        // all heuristics except TSH 1 and TSH 4 //
                if (there is an improvement in total saving)
                {
                        in_iter_w/o_improvement = 0
                        update best solution inside
                }
                Else
                {
                        in_iter_w/o_improvement = in_iter_w/o_improvement + 1
                }
        } while (in_iter_w/o_improvement < IIT)
        Identify the new restart using IN_LTM frequency matrix        // all heuristics except TSH 1 and TSH 4 //
} while (the number of restart < 2)

Return: the best part option solution and corresponding total savings
```