AN ABSTRACT OF THE THESIS OF

Yunan Karim for the degree of Master of Science in Industrial Engineering presented on December 9, 1999. Title: The Impact of Alternative Cell Locations and Alternative Routes of Material Handling Equipment in the Design of Cellular Manufacturing Systems.

Abstract approved: _____

Rasaratnam Logendran

Cellular Manufacturing Systems (CMS) have claimed many advantages over traditional job shop processes. Some of the advantages reported by several users of CMS are reduction in throughput time, reduction in WIP inventory, improvement in product quality, faster response time to customer orders, shorter move distances, increase in manufacturing flexibility, and greater job satisfaction. In its implementation, CMS organizes a production floor into manufacturing cells. Hence, the important issue that needs to be addressed first is the cell formation (CF) problem. CF deals with the identification of part families, machine groups, and allocation of part families and machine groups to cells or vice versa. In the past, most studies in CF have assumed that the location of a cell is known a priori and a unique route exists between two cells. However, in an actual manufacturing environment, alternative locations are available for locating each cell. Similarly, when the capacity of the material handler being used is limited, alternative routes may have to be used to move part loads between two cells.

In this research, the issues dealing with alternative cell locations and alternative routes of material handling equipment are investigated. In addition, several other important factors common to CF are also considered. These include machine capacity limitations, batches of part demands, non-consecutive operations of parts, and maximum number of machines assigned to a cell. A mathematical model is first formulated to represent the research problem. The model is a binary and general integer non-linear

programming model, and it belongs to the class of NP-hard problem. Therefore, a higher level heuristic algorithm, based on the concept known as tabu search, is developed to efficiently solve the problems with industry merit. Incorporating the features associated with the tabu search, resulted in developing six different versions of the heuristic solution algorithm. The six heuristics are tested on twenty small problems, and the quality of their solutions is evaluated by investing significant effort to find their optimal solutions. The evaluation shows that the heuristics are highly effective. The solutions obtained from the heuristics have average percentage deviation of less than 3% from the optimal solutions. The heuristics are also tested on their performances with medium and large problems.

By using a statistical experiment that is based on randomized block design, the performance of the six heuristics is compared. Three different problem structures, ranging from 4 parts to 30 parts and from 3 locations to 9 locations are used in the experimentation. The experiment reveals that in general, the tabu search based-heuristic using fixed tabu list size and long-term memory based on minimal frequency strategy is preferred to other heuristics as the problem size increases.

The Impact of Alternative Cell Locations and Alternative Routes of Material Handling Equipment in the Design of Cellular Manufacturing Systems

by

Yunan Karim

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented December 9, 1999
Commencement June 2000

Master of Science thesis of Yunan Karim presented on December 9, 1999.

APPROVED:

Redacted for privacy

Major Professor, representing Industrial Engineering

Redacted for privacy

Head 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

Yunan Karim, Author

ACKNOWLEDGEMENT

A great many people have made significant contributions to this thesis. They have been such an encouragement to me that I appreciate each of them very much.

My greatest gratitude goes to my wife, Pricillia Halim, who has continually supported and encouraged me to keep working on this thesis. She has been very understanding and patient with me. She endured all the inconveniences that I caused while working on the thesis.

My sincere thanks go to my Major Professor, Dr. Rasaratnam Logendran, who has continually advised me throughout the pursuit of my research. His knowledge in the Cellular Manufacturing Systems has inspired me tremendously about this subject. Above all, his commitment to the completion of this thesis has challenged me to work hard and persevere.

I would like to thank the faculty of the Industrial and Manufacturing Engineering (IME) Department for providing me with an assistantship during part of my graduate study at Oregon State University. I am also grateful to the staff of IME, especially Yenny Djajalaksana, who helped me in setting up the computers for carrying out the research.

I would also thank my committee members: Dr. Ken Funk, my Minor Professor, Dr. Kimberly Douglas, my committee member, and Dr. James E. Reeb, my graduate council representative, for taking the time to evaluate my thesis.

Finally, I would like to thank all my friends for their support and encouragement. Richard Lim, Dave and Ani Pearman, and Steve Glick for motivating me repeatedly to finish the thesis. Zhang Zhongming and Fenny Subur for the information they shared in computer programming and experimentation.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

Figure Page

LIST OF TABLES

LIST OF TABLES (CONTINUED)

LIST OF APPENDICES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX TABLES

Dedicated to

Jesus Christ

and

My Parents

THE IMPACT OF ALTERNATIVE CELL LOCATIONS AND ALTERNATIVE ROUTES OF MATERIAL HANDLING EQUIPMENT IN THE DESIGN OF CELLULAR MANUFACTURING SYSTEMS

1. INTRODUCTION

Due to the increasing demand to improve manufacturing flexibility, product quality, and inventory lead times, manufacturers have searched for ways to move beyond the traditional manufacturing system that employs either a production line or a job shop process. Manufacturers strive to get the mutual benefits from both systems in order to meet the rapidly changing demands from customers and for market competitiveness. Group Technology (GT) has been found to be particularly promising for those manufacturers. GT has been reported to possess some of the strategic benefits of a job shop process such as product customization and also some of the operational benefits of a production line such as reduced WIP inventories (Hyer 1984).

Group Technology is defined as a manufacturing philosophy that seeks to take advantage of the design and processing similarities among parts. One application of GT is the Cellular Manufacturing System (CMS). CMS focuses on grouping parts that have similar processing requirements into part families and then assigning the machines that are required by those part families into manufacturing cells. When forming these production cells, its main objective is to achieve a set of independent cells having the least amount of interactions with one another. CMS has claimed many advantages over the traditional job shop process. Some of the advantages that can be directly attributed to forming manufacturing cells are (Gaither 1992):

1. Machine changeovers between batches of parts are simplified, thereby reducing cost of changeovers and increasing production capacity.
2. Variability of tasks is reduced, and training periods for workers are shortened.
3. Direct routes through production are improved, allowing parts to be produced faster and shipped quicker. As a result, in-process inventory levels will also drop.

4. Workers are specifically trained for a cell so their responsibilities are more focussed and quality control is improved.
5. With shorter and more direct routes, material handling costs are reduced and production and planning controls are simplified.
6. With a cell dedicated to specific part families, the tooling and machines required for a cell is simplified. This process of simplification within a cell facilitates the automation of a complex job shop by dealing with the automation of one cell at a time. Therefore, the formation of cells may be seen as an intermediate step in the automation of job shops.

Several users of CMS have also reported the benefits they received from its implementation. From a study of 32 firms (Wemmerlov and Hyer 1989), the range of percentage savings reported by US manufacturers are setup cost decreased 20-60%; labor costs decreased 15-25%; tooling cost decreased 20-30%; rework/scrap decreased 15-75%; machine tools costs decreased 15-25%; and WIP costs decreased 20-50%. In another study (Ransom 1972), a company reported a 32 % increase in sales, a 44% decrease in overall inventory and an 83 % decrease in the shipment of late orders. Certain intangible benefits are also reported such as the increase in customer satisfaction (Askin and Subramaniam 1987).

In implementing CMS, the first problem that needs to be addressed is the cell formation (CF) problem. CF deals with the identification of part families, machine groups, and allocation of families to cells or vice versa. During the past two decades, a considerable amount of research has been devoted to this issue. Most researchers included one or several objectives in their investigations. The single objective of CF research (McAuley 1972 and Carrie 1973) has been considered as impractical simply because it is not able to satisfy some of the other CF objectives. On the other hand, the research that focussed on multiple objectives of CF has received considerable attention. It employed many different kinds of techniques including heuristics, mathematical programming, artificial intelligence, and combinatorial search. Classification of the techniques used in CF has also been proposed by several researchers (Burbidge 1979; Wemmerlov and Hyer 1986; Ballakur and Steudel 1987; Moodie et al. 1995; Selim et al 1998).

This research attempts to address several objectives that are of major importance to the design of CMS by employing the combinatorial search technique called tabu search to its procedure. It is motivated by the lack of investigation from past CF researchers in the area of alternative cell locations and the role of material handling equipment during the formation of cells. First of all, most CF researchers assumed that the location of the individual cell is fixed or known ahead. They focussed on assigning machines to cells and neglected the flexibility of placing cells in different locations. Therefore, by removing this assumption, a better cell formation could be obtained by investigating the right location for each cell. Secondly, previous CF researchers have never considered the role of material handling equipment in the formation of cell. They assumed that only a single route exists between any two cells. This assumption is valid if the capacity of the material handling equipment is always unlimited. If such were the case, only the shortest route between any two cells would be considered. However, in any real manufacturing system, the capacity of the material handling equipment (e.g. carts, forklifts, AGVs) can be limited due to changes in demand, process delays or equipment breakdown. When the capacity of the material handling equipment becomes limited, alternative routes in transporting part loads have to be considered in addition to the shortest route. These alternative routes require that the research be performed to include the interaction of material handling equipment in the investigation of the CF problem.

Beside the two major issues described in the previous paragraph, this research includes several other factors that are of importance to the design of CMS. These include machine capacity limitations, batches of part demands, non-consecutive operations of parts, and maximum number of machines assigned to a cell. By including all these factors, this research aims at providing a comprehensive framework to the design of CMS. First, the CF problem is investigated from the mathematical programming viewpoint. A mathematical model is developed to include all the factors described earlier. Using the insight gained from the model, heuristic algorithms are developed to solve the CF problem.

Including this chapter, this thesis contains eight chapters. The remaining chapters are organized as follows. Chapter 2 reviews several approaches used in solving the CF problem. Chapter 3 presents major CF issues considered in this research. Chapter 4

describes the mathematical formulation used in representing the CF problem. In Chapter 5, the proposed heuristic algorithms and its application are presented in detail. In Chapter 6, the procedure used in assessing the quality of the heuristic algorithms is given. Chapter 7 presents the experimental design used in comparing the different heuristic algorithms developed in this research. Finally, conclusions and recommendations for future research are presented in Chapter 8.

2. LITERATURE REVIEW

The concept of Group Technology (GT) was first applied in Soviet Union by Mitrofanov and Sokolovskii in the late 1940s. Since then, it has received considerable attention from both academics and industry practitioners for the past two decades. The recent development of GT has followed three basic stages for its industry applications (Askin and Vakharia 1990). The first stage involves the development of a part coding scheme. The second stage involves cell formation. Finally, the third stage involves specifying the machine layout within cells. Among these three stages, the second stage has received the most recognition from researchers and has developed into a distinct field of research called the Cellular Manufacturing System (CMS).

The CMS organizes a production floor into manufacturing cells. Each cell serves as a production unit for which a group of functionally dissimilar equipment are located in close proximity and dedicated to manufacture a family of parts or products with similar characteristics (Burbidge 1979). Some of the main objectives sought in implementing the CMS are to reduce throughput time, reduce WIP inventory, improve product quality, reduce response time to customer orders, reduce move distances, increase manufacturing flexibility, reduce inventories, and increase job satisfaction (Wemmerlov and Johnson 1996). Several users of the CMS in various fields have also reported the benefits they received from its application. Several researchers have also studied the performance of the CMS in terms of percentage of cost savings and percentage of time improvement (Wemmerlov and Hyer 1989, Harvey 1993, Wemmerlov and Johnson 1996).

With the numerous benefits that CMS could offer, it is apparent that it would attract many researchers into this area. Indeed, in the past two decades there have been many analytical methodologies developed to solve the cell formation problems. All these methodologies rely on the routing or process plan of the parts to initiate the grouping process. Having this relationship between parts and machines, the cell formation would be able to identify the part families and the group of machines on which these parts are to be processed. Thus, basically at this cell formation stage, there are three major decisions that would need to be made, which are the identification of part families, the

identification of production cells, and the allocation of the families to cells or vice versa. While making these decisions, the operational constraints will also be taken into consideration. The constraints that are of interest to researchers are (Vakharia and Wemmerlov 1990):

1. Minimize the intercell material handling costs or maximize the cell independence.
2. Minimize the investment in equipment
3. Maintain acceptable equipment utilization levels
4. Identify cells of a reasonable size.

Over the years, several researchers have proposed classification of methodologies used in cell formation (Burbidge 1979; Wemmerlov and Hyer 1986; Ballakur and Steudel 1987; Selim et al. 1995; Moodie et al. 1995). These classifications are useful to the researchers by providing them with an overall understanding of the cell formation scope. A brief overview of the latest classification published at the time of this research (Selim et al. 1998) is presented below. In this classification, the work associated with cell formation was divided into five categories: descriptive procedures, cluster analysis, graph partitioning, artificial intelligence, and mathematical programming. A short description for each category is as follows:

1. Descriptive Procedures.

   In general, this category is broken into three major classes. The first class, which is referred to as part families identification (PFI), begins the cell formation process by identifying the part families and then allocates machines to the families. In identifying the part families, the procedure could be as simple as visual examination or rules of thumb to more reliable techniques such as coding and classification system (Wemmerlov and Hyer 1986). The second class, which is referred to as machine groups identification (MGI), begins the cell formation with the reversal steps of the first class. The MGI procedure is performed into two stages. In the first stage of analysis, machines are grouped based on part routings information, and then in the second stage, parts are allocated to machine groups. The third class attempts to perform the tasks of the first and the second class simultaneously. This approach was first proposed by Burbidge in 1963, which is referred to as Production Flow Analysis (PFA). PFA used the operation sequence and machine routing to group parts, which

have similar processing characteristics. At the same time, another approach in the same class was also proposed by El-Essawy in 1971, which is referred to as Component Flow Analysis (CFA). CFA used the information from the part mix as a starting basis for forming groups rather than dividing the shop into major groups. There was some amount of subjectivity involved in the cell formation stages in this approach as well as the PFA approach.

2. Procedures based on cluster analysis.

   Cluster analysis method attempts to identify structure in a complex data set. Its main objective is to group either objects or entities or their attributes into clusters such that individual elements within a cluster have a high degree of "natural association" among themselves and that there is very little "natural association" between clusters.

3. Graph Partitioning approaches.

   Graph partitioning method treats the machines or parts as vertices and the processing of parts as arcs connecting these nodes. These models aim at obtaining disconnected subgraphs from a machine-machine or machine-part graph to identify manufacturing cells.

4. Artificial Intelligence approaches.

   Artificial Intelligence method uses domain specific knowledge rules and a prototype feature based on modeling system to automate the process of identifying parts attributes and assigning the parts to the most appropriate manufacturing cell. The expert assignment system is based on the geometric features of the parts, characteristics of formed manufacturing cells, parts' functional characteristics and attributes, as well as domain specific manufacturing knowledge.

5. Mathematical programming approaches.

   Mathematical programming methods can be further classified into five major groups based on the type of formulation: linear programming (LP); linear and quadratic programming (LDP); dynamic programming (DP); non-linear programming (NLP); and goal programming (GP)

The development of mathematical programming in the cell formation field has emerged as one of the most promising approaches. However, most of the mathematical models developed by this approach are computationally intractable for large problems.

Most of them can only handle small problems and they were impractical for industry applications. In order to simulate the capability of mathematical programming, researchers have developed search heuristics. This search heuristic attempts to find quality solutions at a reasonable computational cost. With a strategic and intelligent application, the search heuristics can prove to be very efficient and effective methods. Search heuristics have been used many years in various fields of study and have been reported to be very effective in solving large problems. The most recent development of search heuristics is in the area of simulated annealing, tabu search, genetic algorithms, and artificial neural networks. The application of search heuristics in solving the cell formation problems is not considered new anymore. Simulated annealing was used by Boctor (1990); Venugopal and Narendran (1992); and Chen and Srivstava (1994). Tabu search was used by Logendran et al. (1994); Logendran and Ramakrishna (1995, 1996, and 1997); Vakharia and Chang (1997); and Aljaber et al. (1997). Genetic algorithms were used by Venugopal and Narendran (1992). Neural networks were used by Javed (1993).

The research reported here employs the tabu search to solve the cell formation problem. Tabu search is a higher level heuristic procedure for solving optimization problems, which is designed to guide other methods (or their component processes) to escape the trap of local optimality (Glover 1990). It was pioneered by Fred Glover (1986) and presented in detail in Glover (1989,1990), and Glover and Laguna (1992). Tabu search has been used to obtain optimal and near optimal solutions for a wide variety of applications. Some applications of tabu search have included scheduling, transportation network design, layout and circuit design problems, telecommunications, probabilistic logic and expert systems, neural network pattern recognition, and others (for a list of references and a brief exposition of such application papers, see Glover and Laguna 1992). Although it is still in an early stage of development, tabu search has enjoyed a number of successes. In a variety of problem settings mentioned above, it has found solutions superior to the best previously obtained by alternative methods.

As previously mentioned, at the time of this research there are three papers published regarding the use of tabu search in the cell formation. Each paper describes a procedure that differs from others based on the approach to the cell formation problem,

the objectives and constraints that are used to guide the search, and the tabu search features that are employed in the search. In the following paragraphs, a brief overview of each tabu search heuristic developed by early researchers is presented.

1. Logendran and Ramakrishna (1995, 1996, and 1997) approached the CF problem by first considering the parts and machines assignment, and then grouping machines to form cells. Their objective is to minimize the total number of moves (intercell and intracell) required to transport all the loads. In their research, the tabu search performs the tasks of assigning a part to a machine and forming cells simultaneously by means of perturbation. Their implementation of tabu search has only considered the short-term memory structure.
2. Aljaber et al. (1997) approached the CF problem by identifying part-machine clusters, in a part-machine matrix, with the objective of minimizing the total number of intercell moves. The procedure starts by first solving two Shortest Spanning Path (SSP) problems, one for parts (columns) and one for machines (rows). Then the resulting spanning paths for parts and machines were decomposed into subgraphs that represent machine groups and part families, respectively. In their research, the tabu search is used to identify new SSP sequences of parts and machines by means of adjacent pairwise interchange, insert, and swap. Their implementation of tabu search has also considered the short-term memory structure only.
3. Vakharia and Chang (1997) approached the CF problem by first developing a mathematical model to the cell formation problem with the objective of minimizing the total cost of equipment used to identify the manufacturing cells. The procedure starts with a randomly generated initial solution that satisfies the constraints required by the mathematical model. Finally, the tabu search is used to identify new solution by means of interchanging part operations. Their implementation of tabu search has also considered the short-term memory structure only.

The distinguishing feature of this research is not so much on the use of tabu search in solving the cell formation problem even though this research has employed the more sophisticated features of tabu search. This research distinctly addresses the role of alternative cell locations and the role of material handling equipment (specifically the alternative routes of material handling) in cell formation. In all the studies conducted in

cell formation, it is always assumed that the location of the individual cell is fixed or known ahead. Past researchers have assumed that the assignment of cells to locations is performed after the process of cell formation is completed. This assumption would lead to losing a degree of flexibility that could help in finding a better solution. Furthermore, previous studies never considered the alternative routes of material handling equipment. Researchers assumed that the movement of parts between cells could be accounted for in the model using a single distance measure. However, in a real manufacturing system, when the capacity of the material handling equipment can be limited, alternative routes of transporting part loads have to be considered. In this research, therefore, not only the location of a cell is considered flexible but also the routing of material handling equipment.

In modern manufacturing systems where Cellular Manufacturing System (CMS) and Flexible Manufacturing System (FMS) are concurrently applied, the role of automated material handling systems such as the Automated Guided Vehicle (AGV) becomes very important. The AGV offers a high degree of routing flexibility and it can be integrated with computerized systems. In addition, the flow path of AGV proves to be more flexible and reliable than other material handling equipment such as conveyors. There are two types of AGV distribution systems, namely the conventional network AGV system and the tandem configuration AGV system. In a conventional AGV system, multiple vehicles exist and each one is allowed to serve any station in the system. Implementation of such a system requires careful consideration of several important issues such as flow path design (Gaskins and Tanchoco 1987), number of vehicles required (Maxwell and Muckstadt 1982), location of pickup and delivery stations (Goetz and Egbelu 1990), vehicle management and routing (Egbelu and Tanchoco 1984), choice of vehicle types (Leung et al 1987), and traffic management (Malmborg 1990). On the other hand, a tandem configuration AGV system is based on partitioning of all stations into non-overlapping, single vehicle closed loops with the additional transit points provided as interfaces between adjacent loops (Bozer and Srinivasan 1989, 1991, 1992, Lin et al. 1993). Each transit point has two transit stations to transport loads bi-directionally between adjacent loops. The advantages of the tandem AGV system over the conventional AGV system include less complicated vehicle dispatching and traffic

management, elimination of congestion and conflict, facilitating distributed control, and offering expansion flexibility. A more detailed discussion has been given by Bozer and Srinivasan (1989, 1991). Because of the benefits that tandem AGV can offer over the conventional AGV and coupled with the structure of tandem AGV that conforms to the CMS environment, the tandem AGV configuration is selected as the choice of material handling system in this research.

3. PROBLEM STATEMENT

Over the past decade, the research on cell formation has progressed significantly. It started in the 1970s when Burbidge proposed the Production Flow Analysis (PFA) to solve the cell formation problem. The PFA approach was heavily manual and involved much subjective judgement. Then came several approaches to cell formation that concentrated on creating independent cells through part or machine matrix manipulation and clustering techniques (King 1980b, Chandrasekharan and Rajagopalan 1986, Kusiak and Chow 1988, McAuley 1972, Carrie 1973, Seifoddini 1989b, and many others). Most of these approaches do not consider either one or more of these important factors such as flow directions, flow volumes, cell layout design, and material handling flexibility. Hence, these approaches fail to reflect the needs of real manufacturing systems. Therefore, several heuristics were developed to solve the cell formation problem, which included some of the factors that were neglected by previous researchers. Due to the complexity of the cell formation problem, most heuristics are not able to account for all of the factors that would affect the design of a cell. Recent approaches attempt to address several factors that are of major importance to the cell formation problem. These approaches have provided valuable insights to the cell formation issues. They have also laid the foundation for the future development of a comprehensive cell formation approach.

The research undertaken here is partly motivated by the finding from the previous research (Logendran et al.1994; Logendran and Ramakrishna 1995, 1996, and 1997; Puvanunt 1996; Logendran and Puvanunt 1997), where tabu search had been successfully used to identify a quality solution to a variety of problems in cellular manufacturing systems. Tabu search is a metaheuristic that can be superimposed on other procedures to prevent them from becoming trapped at locally optimal solutions. Besides the success of tabu search in identifying a quality solution, it has the ability to start with a simple implementation that can be upgraded over time to incorporate more advanced or specialized elements. This upgrade capability of tabu search has allowed researchers to incorporate more design factors into their cell formation heuristics. This research has also

made use of some of the insightful finding from the previous research in tabu search-based applications to the cell formation problem.

In this research, some of the design factors included in the previous research are taken into consideration. These include machine capacity limitations, batches of part demands, non-consecutive operations of parts, and the maximum number of machines assigned to a cell. In addition, this research uniquely identifies the role of alternative cell locations and alternative routing of material handling equipment as two significant design factors in the investigation of the cell formation problem. Previous research in CMS has always assumed that location of individual cells is fixed. There are two reasons for making this assumption. The first reason is that previous research was more focussed on creating independent cells, thus reducing inter-cellular movements. The second reason is that the location of individual cells can be determined after the cells are formed. The argument against the first reason is that having less inter-cellular movements does not necessarily translate into efficient use of material handling equipment. In creating the independent cells, previous research has disregarded the flow volume factor. Disregarding the flow volume factor could lead to inefficient use of material handling equipment. As a result the processing time of parts as well as the cost of moving parts will increase. The argument against the second reason is that if the location of individual cells is determined after the cells are formed, a degree of flexibility that could help in attaining a better solution may be lost. This research focuses on simultaneously dealing with cell formation and location identification. For the aforementioned reasons, it is of major importance to include alternative locations for individual cells as a significant design factor in the investigation of CMS.

In any manufacturing system including CMS, material handling is one of the important factors that merits consideration. The types of material handling equipment used could be carts, forklifts, conveyors, or AGVs. In the previous research of CMS, the interaction between material handling equipment was never considered as one of the important factors in the design of cell. Previous work has assumed that there is only a single route that exists for movement between any two cells. For example, movement between any two cells will always use the shortest route between the two. The material handlers used to perform the movements have never come into the picture because it is

assumed that they are always available or their capacity is unlimited. Therefore, there is no need to consider the complexity of managing the movements between any two cells. Notice that in transporting loads between any two cells, several material handlers could work together to accomplish the task, rather than the common presumption that only one material handler is adequate to perform the task. Therefore, if the capacity of the material handlers is disregarded during the formation of a cell, this interaction of material handlers can also be disregarded. The cell formation would only need to find the shortest route between any two cells. However, if the capacity of the material handlers is limited, the interaction of material handlers might reveal that several alternative routes should be considered rather than just the shortest route. Realistically, the capacity of a material handler is limited. Under these circumstances, the coverage needed over and above a material handler's capacity would have to be provided by other handlers included in the system. Thus, the interaction between material handlers and the existence of multiple routes of movement between cells become important issues. In the following example, a detailed description of the situation, which may arise in any manufacturing system, is presented. The choice of the material handler used in this example and throughout the research is Automated Guided Vehicle (AGV).

In Figure 3.1, three AGVs are used to coordinate material transfers between the Input/Output (I/O) and any of the six locations. Notation L will be used to denote location. AGV 1 is dedicated to coordinate material transfers on the loop covering I/O, L1 and L2; AGV 2 on the loop covering L3 and L4; and AGV 3 on the loop covering L5 and L6. Between adjacent loops, there are transfer points T1, T2, and T3. These transfer points serve as the buffer for movement in and out of the loops. In the operation of AGVs, a terminal point for each AGV must be assumed. The terminal point for an AGV is the location where the AGV is held when it is not in use. In this example, I/O, L3, and L6 are assumed to be the terminal point for AGV 1, 2, and 3, respectively. If there is an order to move parts from L1 to L3, then logically, AGV 1 will move from I/O to L1, load the parts, bring it to T1, unload the parts and then return back to its terminal point I/O. Next, AGV 2 will have to move from L3 to T1, load the parts, bring it to L3, unload the parts and the process is completed because L3 is the terminal point for AGV 2. The route that has just been taken is the shortest route from L1 to L3. However, if during the

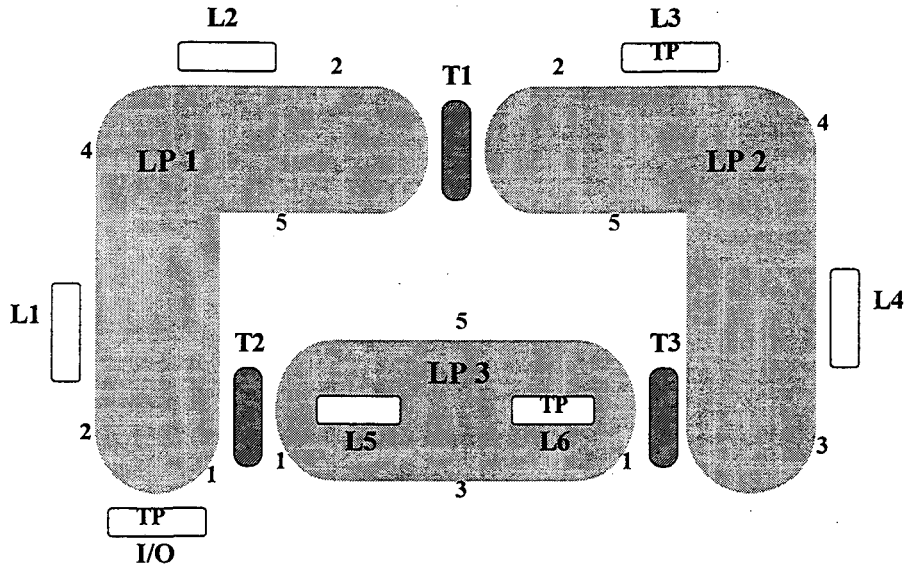

Figure 3.1 AGV system based on tandem arrangement

production horizon, the capacity of AGV 1 has become inadequate, then alternative route should be provided to cover the needs to move from L1 to L3. In this case, instead of taking the parts from L1 to T1, AGV 1 can take the parts from L1 to T2, which will save considerable AGV 1 usage time. Then, AGV 3 has to take the parts from T2 to T3, and AGV 2 has to take the parts from T3 to L3. This second route is certainly more expensive than the first route in term of travel distance and AGV usage time. Its travel distance is longer and it takes more of AGV 2 and AGV 3 usage time than the first route, but it is justified because there is just not enough capacity in AGV 1. The first route, which is the shortest route from L1 to L3 costs 14 distance units on AGV 1, while the second route costs only 2 distance units on AGV 1. Notice also that this second route is feasible as long as the capacity of AGV 2 and AGV 3 is not violated. In relation to the cell formation research, the manner in which a cell is formed and located will be affected by the presence of material handling equipment in the manufacturing system.

Clearly, alternative cell locations and alternative routing of material handling equipment are two most significant factors that must be considered in the investigation of the cell formation problem. Consequently, the objectives of this research can be stated as:

(i) To develop a mathematical model that is capable of addressing the needs of a cellular manufacturing system in the presence of design factors including machine capacity limitations, batches of part demands, non consecutive operations, maximum number of machines assigned to a cell, alternative locations for an individual cell, and alternative routes for AGVs and their capacity limitations.

(ii) To develop an efficient solution algorithm that can be used to solve the model specified in item (i). The algorithm should be capable of identifying a quality solution within a reasonable computational time, even on industrial-size problems.

In the next chapter, the mathematical model for this problem is formulated as a polynomial programming model. Its objective function focuses on minimizing the total service time required to meet the production demands. The service time is defined as the time taken to perform a material transfer from the time an order is received until the time the material is delivered to its destination.

4. MODEL DEVELOPMENT

4.1 Background

The model developed in this research uniquely addresses the issues of alternative cell locations and alternative routing of material handling equipment in CMS. It is formulated as a polynomial programming model. The objective function focuses on minimizing the total time required in transporting all the part loads during a planning horizon. The constraints consist of equations that deal with the major issues described in Chapter 3.

The assumptions and notations used in the development of an appropriate mathematical model are stated below. Following this, a mathematical model, which includes the objective function and constraints, is presented. Finally, the description of the model as well as its computational complexity is also given.

4.2 Assumptions

(1) The cellular movements within a cell are negligible. In this research, AGVs are used to coordinate material transfers between cells but not within a cell. Thus, the movements within a cell do not consume any AGV usage time and hence, it is not considered in the mathematical model. As a matter of fact, including the cellular movements within a cell into a model is not necessary because in solving the CF problem, it is the movement between cells that the model tries to minimize and not the movements within a cell. Therefore, including the cellular movements within a cell will just complicate the model without adding any benefits to it.

(2) The planning horizon of production is one day.

(3) The annual demand for each part type is known and stable over the planning horizon.

(4) The multiple routes of material handling equipment are specified. It means that the routes between any two cells are pre-determined.

(5) The available capacity of each unit of a machine type is known.

(6) The available capacity of each unit of AGV is known.

(7) For each part, the time and machine required to perform a specific operation is known.

4.3 Notations

i = 1,..., m machine types

p = 1,..., m_i unit of machine type i

j = 1,..., n part types

r = 0, 1,..., c locations (0 representing I/O)

k = 1,..., k_j operations for part j

g = 1,..., G AGVs

rt = 1,..., $rt(r_1,r_2)$ routes between location r_1 and r_2

$$S(j,k,p,r) = \begin{cases} 1 & \text{if part } j\text{'s kth operation is performed on unit machine } p \text{ in location } r \\ 0 & \text{otherwise} \end{cases}$$

$$X_{ipr} = \begin{cases} 1 & \text{if unit machine } p \text{ of type } i \text{ is assigned to location } r \\ 0 & \text{otherwise} \end{cases}$$

p_{jki} = Total processing time (workload) for performing operation k of part j on machine type i over the planning horizon

b_j = Number of batches of part j produced over the planning horizon

k_j = Number of operations scheduled to be performed on part j

m_i = Number of machines type i required

n_r = Maximum number of machines that can be assigned to any location

C_{ip} = Available capacity on machine p of type i over the planning horizon

MM = Set of machine types having two or more units, i.e, $m_j>1$ for i ∈ MM

PP = Set of parts requiring two or more operations be performed, i.e., $k_j>1$ for j ∈ PP

$rt(r_1,r_2)$ = Number of possible routes between locations r_1 and r_2

$F(r_1,r_2,rt)$ = Total service time between locations r_1 and r_2 using route *rt*

B(j,k,rt) = Number of batches of part *j* transported using route *rt* during operation *k*

$TT(g,r_1,r_2,rt)$ = Total travel time required for AGV *g* to move from location r_1 to r_2 using *rt*

A_g = Available capacity available on AGV *g* during the planning horizon

4.4 Mathematical Model

Minimize

$$Z = \sum_{j=1}^{n} \left[\begin{array}{l} \sum_{r=1}^{c} \sum_{p=1}^{m_i} \sum_{rt=1}^{rt(0,r)} S(j,1,p,r)\ F(0,r,rt)\ B(j,1,rt) + \\ \sum_{\substack{k=1 \\ \text{and} \\ j\in PP}}^{k_j-1} \sum_{r_1=1}^{c} \sum_{p=1}^{m_i} S(j,k,p,r_1) \left\{ \sum_{r_2=1}^{c} \sum_{p=1}^{m_i} \sum_{rt=1}^{rt(r_1,r_2)} S(j,k+1,p,r_2)\ F(r_1,r_2,rt)\ B(j,k+1,rt) \right\} + \\ \sum_{r=1}^{c} \sum_{p=1}^{m_i} \sum_{rt=1}^{rt(r,0)} S(j,k_j,p,r)\ F(r,0,rt)\ B(j,k_j+1,rt) \end{array} \right]$$

subject to:

$$\sum_{r=1}^{c}\sum_{p=1}^{m_i} S(j,k,p,r)=1 \qquad ;\begin{matrix} j=1,\ldots,n \\ k=1,\ldots,k_j \end{matrix} \qquad (1)$$

$$\sum_{r=1}^{c} X_{ipr}=1 \qquad ;\begin{matrix} i=1,\ldots,m \\ p=1,\ldots,m_i \end{matrix} \qquad (2)$$

$$C_{ip}\, X_{ipr} \geq \sum_{j=1}^{n}\sum_{k=1}^{k_j} p_{jki}\, S(j,k,p,r) \qquad \begin{matrix} r=1,\ldots,c \\ ;i=1,\ldots,m \\ p=1,\ldots,m_i \end{matrix} \qquad (3)$$

$$\sum_{i=1}^{m}\sum_{p=1}^{m_i} X_{ipr} \leq n_r \qquad ;r=1,\ldots,c \qquad (4)$$

$$\sum_{r=1}^{c}\sum_{p=1}^{m_i}\sum_{rt=1}^{rt(0,r)} S(j,1,p,r)\, B(j,1,rt)=b_j \qquad ;j=1,\ldots,n \qquad (5)$$

$$\sum_{r1=1}^{c}\sum_{p=1}^{m_i} S(j,k,p,n)\left\{\sum_{r2=1}^{c}\sum_{p=1}^{mi}\sum_{rt=1}^{rt(r1,r2)} S(j,k+1,p,r2)\, B(j,k+1,rt)\right\}=b_j \qquad \begin{matrix} j=1,\ldots,n \\ ;k=1,\ldots,k_j-1 \end{matrix}$$

$$\sum_{r=1}^{c}\sum_{p=1}^{m_i}\sum_{rt=1}^{rt(r,0)} S(j,k_j,p,r)\, B(j,k_j+1,rt)=b_j \qquad ;j=1,\ldots,n$$

$$\sum_{j=1}^{n}\left[\begin{array}{l} \displaystyle\sum_{r=1}^{c}\sum_{p=1}^{m_i}\sum_{rt=1}^{rt(0,r)} S(j,1,p,r)\, TT(g,0,r,rt)\, B(j,1,rt)+ \\ \displaystyle\sum_{k=1}^{k_j-1}\sum_{r_1=1}^{c}\sum_{p=1}^{m_i} S(j,k,p,r_1)\left\{\sum_{r_2=1}^{c}\sum_{p=1}^{mi}\sum_{rt=1}^{rt(r_1,r_2)} S(j,k+1,p,r_2)\, TT(g,r_1,r_2,rt)\, B(j,k+1,rt)\right\}+ \\ \displaystyle\sum_{r=1}^{c}\sum_{p=1}^{m_i}\sum_{rt=1}^{rt(r,0)} S(j,k_j,p,r)\, TT(g,r,0,rt)\, B(j,k_j+1,rt) \end{array}\right] \leq A_g$$

$$;g=1,\ldots,G \qquad (6)$$

4.5 Model Description

The problem is formulated as a binary and general integer non-linear programming model. The objective function shown in the above mathematical model focuses on minimizing the total service time required to meet the production demands. The service time is defined as the time taken to perform a material transfer from the time an order is received until the time the material is delivered to its destination. Thus, the total service time is the sum of each individual time required by every part's movement between locations. The total service time is used here because it appropriately measures the effectiveness of the grouping and the location of machines as well as the coordination of material handling equipment with respect to part orders when their capacities are limited. It is also indirectly used as a throughput measure of the manufacturing process. The smaller the total service time, the quicker is the transportation of parts between locations, and as a result the sooner the parts get completed. The equation for the objective function consists of three terms. The first term evaluates the total service time required to move a part load from I/O to the location where its first operation is performed. The second term evaluates the total service time required to move a part load between two locations where its intermediate operations are performed. An intermediate operation is an operation between the first and the last, including both. The third term evaluates the total service time required to move a part load from the location where its last operation is performed to I/O.

Before proceeding with the description of each constraint, an important insight to the structure of the model is stated. Typically, the model for the cell formation problem that takes into consideration of alternative cell locations would include two constraints: the first ensuring that each cell can only be located in one location and the second ensuring that each location can only take one cell. In other words, the first constraint would deal with the formation of cell and the second constraint would deal with the placement of cell at the right locations. However, during the course of the model development, it has been discovered that the above two constraints could be combined into one constraint of "creating locations". Instead of grouping machines to create the

cells, the grouping of machines is considered directly at the locations. The term "cell" is sort of misleading because it is the contents of the cell, which is a group of machines that is of major importance and not the "cell" in itself. Several machines that are grouped together will define a cell and not vice versa. Thus, the term "cell" can be regarded as that used to create a "virtual" manufacturing cell. If the model determines to place specific machines at a location, then that group of specific machines will be called a cell. Hence, instead of introducing variables that identify a machine or a part in relation to a cell, the model will have variables that identify a machine or a part to a location. This representation has two advantages over its conventional representation. First, it significantly reduces the number of variables that need to be introduced into the mathematical model. Second, it greatly simplifies the formulation of equations for the objective function and constraints.

The constraints of the model can be described as follows:

(i) Constraint equation (1) ensures that each operation of a part is completely carried out on a unit of machine at a location. (i.e. lot splitting is not permitted). Constraint (1) is imposed because of several disadvantages that come from having split lots. The disadvantages are:

   a. Having split lots means that there would be an additional setup incurred on the second machine of the same type to perform the operation required by the split lots. Lot splitting is typically avoided in industry practice due to the extra setup time required on the second machine.

   b. In most industy practices, production managers would ensure that there is excess machine capacity available to meet the production demands (such as in the JIT system, the machine capacity usage is normally around 60%). This excess capacity is justified in order to prepare for machine breakdown, tool failure, changes in customer demand, changes in product's priority, etc. Therefore, lot splitting is a rare occurence in real manufacturing environments.

(ii) Constraint equation (2) ensures that each unit of a machine can only occupy one location. Constraint (2) is basically the integration of two constraints that are

discussed in the preceeding paragraphs. It deals concurrently with the formation of cells and the placement of cells at the locations.

(iii) Constraint (3) ensures that feasible capacity is maintained on every machine. The capacity of each machine is assumed to be 8 hours.

(iv) Constraint (4) ensures that the total number of machines assigned to each location does not exceed the limit imposed. The maximum number of machines assigned to a location is *priori* determined by the management and is dependent upon several factors including flow efficiency, floor space, and number of workers assigned to a cell.

(v) Constraint equation (5) ensures that for each operation of a part, the total number of batches transported through different routes is equal to the number of batches intended to be processed. It is also used to tie the relationships between the movement of a batch and the location where an operation of a part is performed. Notice that each batch of a part can go through different routes to satisfy the part's requirements. Similar to the objective function, the equation for constraint (5) consists of three terms. The first term evaluates the batch movement from I/O to the location where the part's first operation is performed. The second term evaluates the batch movement between the two locations where the part's intermediate operations are performed. An intermediate operation is an operation between the first and the last, including both. The third term evaluates the batch movement from the location where the part's last operation is performed to I/O.

(vi) Constraint (6) ensures that feasible capacity is maintained for every material handler used, which in this case is an AGV. Similar to constraint (5), constraint (6) consists of three terms. The first term evaluates the usage time of AGV g when it is used to move b_j unit loads from I/O to the location, where the first operation of part j is performed. The second term evaluates the usage time of AGV g when it is used to move b_j unit loads between the two locations where the intermediate operations of part j are performed. Again, an intermediate operation is an operation between the first and the last, including both. The third term evaluates the usage time of AGV g when it is used to move b_j unit loads from the location where the last operation of part j is performed to I/O.

4.6 Computational Complexity of the Research Problem

The mathematical model is formulated as a binary and general integer non-linear programming model. It has a non-linear objective function and a combination of linear and non-linear constraints. Constraint (1) to (4) are linear, while constraint (5) and (6) are non-linear. In general, non-linear programming problems belong to a class of NP-complete problems (Garey and Johnson 1979). This claim alone is not sufficient to conclude that the research problem is an NP-complete problem. A special case of this problem has been investigated by Logendran and Ramakrishna (1995), and proven NP-hard in the strong sense. Their investigation focussed on a cell formation problem that did not include alternative cell locations and alternative routing of material handling equipment. As the special case of this research problem was proven NP-hard in the strong sense, this research problem must be strongly NP-hard as well.

For solving any NP-hard problem, the use of an implicit search algorithm such as the branch and bound technique is highly discouraged because it would turn out to be too time consuming even for a problem of moderate size. Therefore, a higher-level heuristic algorithm would need to be developed to solve large problems encountered in industry practice. This research focuses on developing efficient heuristic algorithms that employ a very successful search engine known as tabu search. In the next chapter, a detailed description of the heuristic algorihms as well as their applications are presented.

5. HEURISTIC ALGORITHM

5.1 Tabu Search Introduction

To solve the cell formation problem described by the mathematical model presented in the previous section, a heuristic solution algorithm that employs tabu search as one of its components is developed. Tabu search (TS) is a metaheuristic procedure for solving combinatorial optimization problems. It is an adaptive procedure that can be superimposed on other methods to prevent them from becoming trapped at a locally optimal solution. TS is designed to cross boundaries of local optimality by systematically imposing and releasing constraints to permit exploration to the otherwise forbidden regions. The method was pioneered by Fred Glover (1986) and presented in detail in Glover (1990a, 1990b, 1991), and Glover and Laguna (1992). The applications of tabu search have included scheduling, transportation network design, layout and circuit design problems, telecommunications, probabilistic logic and expert systems, neural network pattern recognition, and others (for a list of references and a brief exposition of such application papers, see Glover and Laguna (1992)).

5.2 Tabu Search Mechanisms

The TS method has referral to a familiar heuristic called the hill-climbing heuristic, which progresses unidirectionally (in descending direction for minimization and in ascending direction for maximization) from an initial feasible solution to a local optimum. The limitation of a hill-climbing procedure in a combinatorial problem setting is that the local optimum obtained at its stopping point, when no improving moves are possible, may not be a global optimum. In contrast to the hill-climbing, TS provides a guide to continue the exploration without becoming confounded by an absence of

improving moves and without falling back into a local optimum from which it previously emerged. The tabu search is founded on three primary themes (Glover, 1990):

(1) the use of flexible attribute based memory structures designed to permit evaluation criteria and historical search information to be exploited more thoroughly than by rigid memory structures or by memoryless systems.

(2) an associated mechanism of control that would constrain and free the search process (embodied in tabu restrictions and aspiration criteria).

(3) the incorporation of memory functions of different time spans, from short term to long term, to implement strategies for intensifying and diversifying the search. Intensification strategies reinforce move combinations and solution features historically found good, while diversification strategies drive the search into new regions.

From an application standpoint, the work of TS as reported by Glover (1990) is briefly summarized below. The presentation of TS method and features has been adapted accordingly to the application of TS in this research or the field of cellular manufacturing system.

Similar to hill climbing procedure, TS starts with an initial solution. The initial solution could be feasible or infeasible, however it is advised to start with a good feasible solution in order to increase the speed of finding better solutions. In contrast to hill climbing, where the immediate improved solution is used as the next move, TS generates a list of candidates moves from an initial solution by applying simple perturbation methods to the initial solution. This step is usually called a neighborhood search. After that, each candidate move in the list is evaluated and the best one is selected as the next move, subject to certain constraints. These constraints, embodied in the tabu restrictions are designed to prevent the reversal and the repetition of certain moves by rendering selected attributes of these moves forbidden or tabu. The primary goal of the tabu restrictions is to permit the method to go beyond points of local optimality while still making high quality moves at each step. Generally, the tabu restrictions are enforced using a list that stores the attributes of restricted or tabu moves, which is normally called the tabu list. The tabu attributes are basically the attributes of previous moves. The tabu list will keep track of those tabu attributes until they are tenure according to the size of

the tabu list used. The rule for the tenurity of an attribute is based on FIFO. Once an attribute is tenure, it is discarded out of the list and the next attribute will be admitted at the end of the list. The choice of the tabu list used in the search will affect the level of intensification process. Increasing the tabu list size will intensify the search since the number of available moves will be restricted. This restriction of available moves will lead the search into a more focused area, which is the area that has provided good moves in the past. In several of the early applications of tabu search, the best tabu list sizes consistently fell in the interval from five to 12. However, most recently, experimentation points that the preferred tabu list sizes lie in intervals related to problem dimension.

When these tabu restrictions are operating in the search, they are counterbalanced by the application of aspiration criteria. The aspiration criteria is simply the condition that overrides the tabu restrictions. In other words, it would allow the forbidden or tabu moves to be performed in the search process when the aspiration criterion was satisfied. A simple, but widely used, type of aspiration criterion is the removal of tabu status of a move if a candidate move yields the best solution found so far.

As the tabu restrictions and aspiration criteria are at work, each new candidate move will have to go through them. Each move will be checked whether it is admissible as a new current move or it will be dropped from further consideration. Once, the best move is selected, it will be admitted into a list called candidate list (CL). Then, a new list of candidate moves will be generated around the new move and the process will be repeated for that new list until a certain criteria is met to terminate the search. The usual terminating criteria could be the maximum number of moves that has been admitted into the index list (IL) or the prescribed number of moves without improving the best solution has been performed. In the latter case, if there is no improvement in the objective function value after a specific number of iterations has been performed the entire search would be terminated. Up to this termination point, the first phase of TS, which is called the short-term memory, is completed. This short-term memory is the core of tabu search. In many of TS applications, the short-term memory component by itself has produced solutions superior to those found by alternative procedures. However in this research, the second phase of TS, which is called the long-term memory will also be employed to enhance the search.

The long-term memory is designed to help the intensification and the diversification of the entire search procedure. In fact, the fundamental elements of intensification and diversification strategies are already present in the short-term memory component of TS. The long-term memory enhances the potential of identifying new starting points in the regions that were not searched in the past. It starts by keeping track the history of the most frequently searched regions and generates a frequency matrix that will keep record of all the information from the previous moves. Then using the maximization or the minimization rules, it will find a starting point that will lead to a region that has been searched rigorously before (for intensification) or a region that has been avoided before (for diversification), respectively. This new starting point will drive the search to investigate the unexplored regions. Once a starting point is identified, the short-term memory process will be applied to the new starting point as the initial solution for the next stage of TS.

5.3 Heuristic Mechanisms

Before describing the detailed steps of the heuristic algorithm developed in this research, an outline of the heuristic algorithm and also the flow chart of the algorithm will be presented. As stated earlier, this research addresses the issues of capacitated AGVs and alternative cell locations in the design of CMS or to be more specific in the context of the cell formation problem. Therefore, the heuristic algorithm would deal with three major components, which are the AGV system, the machine-location identification, and the common issues related to the cell formation problem. This research aims at integrating these components into one unit and solving them jointly rather than breaking them into three levels of the problem. Breaking the problems into sublevels has its disadvantage because it would cause the heuristic to be short sighted. Each time a sublevel problem is created, a degree of information is lost. Therefore, instead of getting the global optimal solution of the total problem, it would only solve for the optimal solution of each sublevel problem. The sum of these sublevel optimal solutions might not

be the global optimal solution sought in the first place. Having said that, an explanation on the role of each component is provided below.

5.3.1 The Component of AGV System

In this research, the tandem AGV system is selected as the choice of material handling system for the CMS. Tandem AGV system consists of several loops and each loop is managed by one AGV. Each loop can cover one to several locations. To transfer loads between loops, the system strategically places transfer points between loops. When a load has to be transferred to a location outside the loop in which it originated, the AGV from its originated loop will just need to move the load to a transfer point located adjacent to the loop. Another AGV from a different loop will pick up the load from the transfer point and carry it to a location or another transfer point depending on the destination of the load. This interloop transfer can possibly cross through several loops before it gets completed. Hence, multiple routes of loops can possibly be taken by the interloop transfer. When the AGV capacity is unlimited, there is only one route that would need to be considered, which is the shortest route. However, when the AGV capacity is limited, multiple routes would need to be considered and they have to be included into the analysis as the alternative routes rather than just using the shortest route all the time. Sometimes, a combination of routes including the shortest route has to be used to transport all the batches. In this research, this AGV load routing problem is formulated as an integer-linear programming model. It will be solved using a commercial integer linear programming solver (LINDO, 1998).

5.3.2 The Component of Machine-Location Identification

As mentioned earlier, the location of machines will have an impact on how the machines are grouped. In identifying the right location for each individual machine, this research applies the tabu search and perturbation method. The tabu search is used to find

good quality solutions effectively and efficiently. It is effective because it is capable of identifying optimal solutions or near optimal solutions. It is efficient because it is able to solve the problems in much shorter time than is required to solve them using conventional methods. The perturbation method is used to generate new machine-location configurations in order for the tabu search to move from one solution state to another. The procedure used by the tabu search and the perturbation will be provided later.

5.3.3 The Component of Common Cell Formation Issues

Some of the common cell formation issues such as the part-machine assignment, maximum number of machines assigned to a location, machine capacity constraints, and non-consecutive operations, would have to be addressed as well. In this research, the issue of maximum number of machines assigned to a location is handled by the machine-location component described above, while the other three issues will be handled by another tabu search and perturbation method, which are independent from the ones used by the machine-location identification component. The functions of the tabu search and perturbation method for this component will be similar to the previous ones except that they are now developed for assigning parts to machine.

At this point, it is easy to see that the heuristic algorithm operates the tabu search and perturbation methods at two levels. The first level, which is called the inside tabu search, deals with the part-machine assignment and its related issues. The second level, which is called the outside tabu search, deals with the machine-location identification and its related issues. Analyzing the process of assigning parts to machines and the process of identifying the location of machines, there is more variety of ways in placing machines to locations rather than in assigning parts to machines. In other words, the number of permutations for identifying the locations of machines is many more in comparison to the number of permutations for assigning parts to machines. Therefore, the machine-location identification has a significantly higher impact on the design of CMS than the part-machine assignment. Consequently, in the development of the heuristic algorithm, the

outside tabu search will serve as the major search while the inside tabu search will serve as the minor search. The outside search will navigate the overall search while the inside search will make fine adjustments for each move that the outside search takes. The final solution for the problem is composed of the solution corresponding to optimal/near-optimal machine-location identification together with the solution corresponding to optimal/near-optimal part-machine assignment. The flow chart shown in Figure 5.1 illustrates the heuristic mechanism incorporated in the tabu search-based procedure. The pseudo code for the heuristic is also provided in Appendix E.

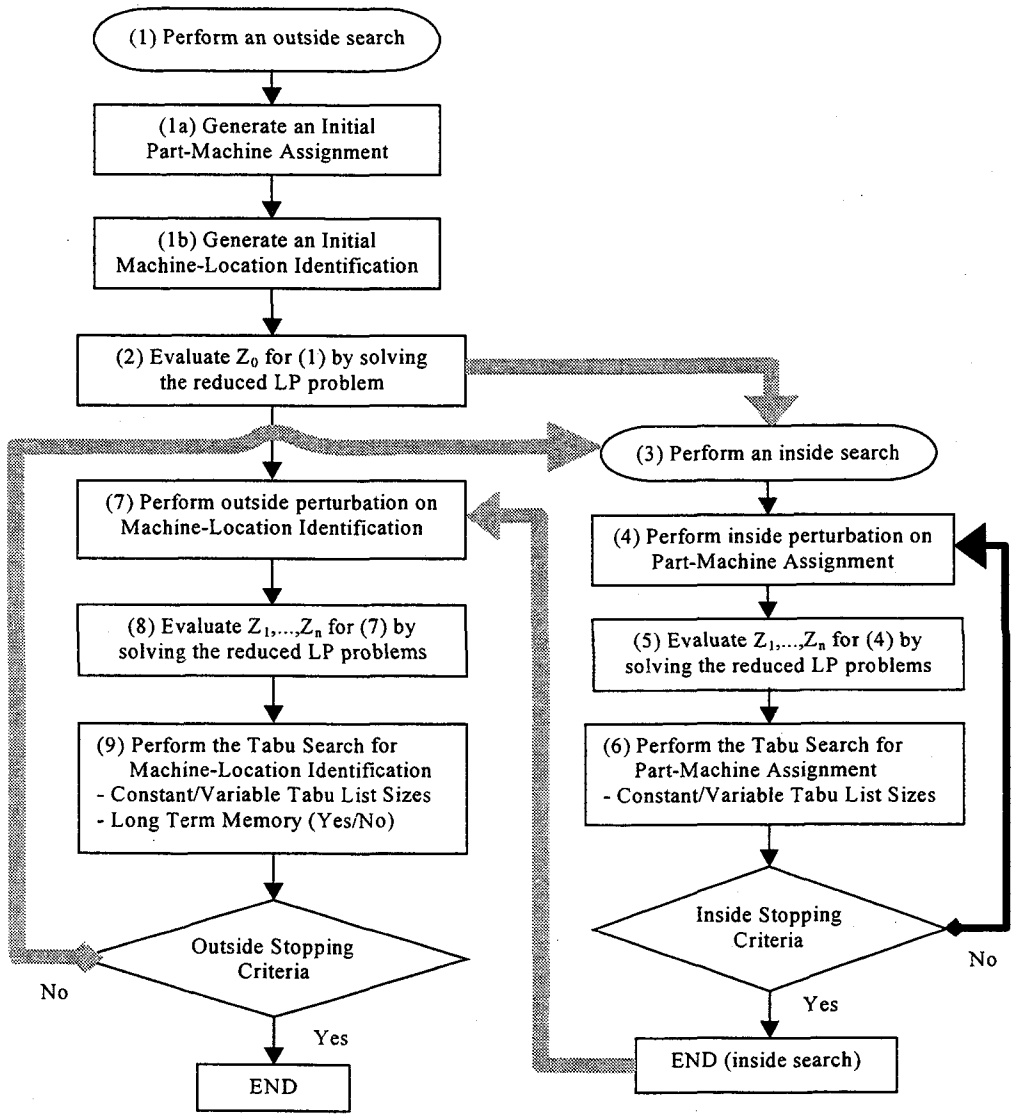

Figure 5.1 Flowchart of tabu search-based heuristic algorithm.

5.4 Steps Associated with Heuristic Algorithm

Notation:

A feasible solution (FS) for the problem considered here consists of a sequence of machine-location identification called FSm and a sequence of part-machine assignment called FSp. For each sequence, a set of seeds can be generated by applying a specific neighborhood function to its current sequence. The two neighborhood functions used for each sequence are defined as follows:

- Nm(FSm) = [FSm': FSm' is a new sequence of machine-location identification obtained from FSm by executing the neighborhood function (Nm)].

  Nm, which is also called the outside perturbation, starts by generating a set of seeds from FSm. Then, it evaluates each seed in the set and returns the one with the minimum value as FSm'.

  The procedure used for the outside perturbation is as follows:

  (1) Perturb on each machine occupying a location, but one machine at a time.

  (2) The perturbation starts by taking a machine occupying a location and inserting it in another location, while the other machines remain at their original locations. If the targeted location where the machine is going to be inserted has reached its maximum number of machines, the perturbation fails and the swapping process would be used.

  (3) The swapping process is performed by exchanging a machine occupying a location with another machine occupying a different location, while the other machines remain at their original locations. Basically, the two machines are swapped, while all the other machines remain at their original locations.

  (4) Perform the outside perturbation on every unique combination of two different locations and for each machine contained in each of the two different locations.

- Np(FSp) = [FSp': FSp' is a new sequence of part-machine assignment obtained from FSp by executing the neighborhood function (Np)].

  Np, which is also called the inside perturbation, starts by generating a set of seeds from FSp. Then, it evaluates each seed in the set and returns the one with the minimum value as FSp'.

The procedure used for the inside perturbation is as follows:

(1) Perturb on each operation of a part performed on a machine type that has multiple units, but one operation of a part at a time.

(2) The perturbation starts by detaching an operation of a part from its current machine and attaching it to another machine of the same type but at a different location, while the other operations remain at their original assignments. If the targeted machine where an operation of a part will be attached to has reached its maximum capacity, the perturbation fails and the swapping process would be used.

(3) The swapping process is performed by exchanging an operation of a part performed on one machine with an operation of a part performed on another machine subject to: both operations are performed on the same machine type; both machines do not have a single job; both machines are not placed at the same location; and the capacity of both machines are not violated. While the swapping process is performed, all the other operations that are not involved in the process will remain at their original assignments.

(4) Perform the inside perturbation on every operation of parts, which can be processed by a machine type that has more than one unit.

Once the notations are given, the steps associated with the heuristic algorithm are ready to be presented. There are all together nine steps; the first step is devoted to generating an initial solution, the second step is used to solve the AGV related issues, the third through the sixth step explain the work performed in the inside search, and finally the seventh through the ninth step explain the work performed in the outside search.

Step 1: Generate the first initial solution (FS_0). FS_0 consists of the first outside initial solution (FSm_0) and the first inside initial solution (FSp_0).

- FSm_0 is the sequence of machine-location identification set by the outside search. It is described as:

  $[L_1:\{m_1, m_2,...,m_X\}; L_2:\{m_1, m_2,...,m_X\};.....;L_Y:\{m_1, m_2,...,m_X\}]$,

where L_i denotes location i, m_j denotes machine j at location i, X is the maximum number of machines that can be assigned to a location, and Y is the total number of locations in the manufacturing system.

- FSp_0 is the sequence of part-machine assignment associated with the present machine-location identification. At the initial phase, it is associated with FSm_0. FSp_0 is described as:

  $[P_1:\{m_1, m_2,...,m_X\}; P_2:\{m_1, m_2,...,m_X\};.....;P_Y:\{m_1, m_2,...,m_X\}]$,

  where P_i denotes part i, m_j denotes the machine used by operation j of part i, X is the number of operations for part i, and Y is the total number of parts in the manufacturing system.

In deciding FSm_0 and FSp_0, the algorithm uses the simple procedure reported by Logendran (1991). The procedure starts by identifying the preliminary part-machine assignment (FSp_0). Then, it groups machines to form cells and assigns each cell to a location sequentially (FSm_0). The description of the algorithm used by the procedure (Logendran, 1991) is given below.

Step 1.1: The steps associated with the part-machine assignment (FSp_0) are as follows:

(i) If a part's operation is performed on a machine type that has only one unit, then that part must be assigned to the unit.

(ii) If a part's operation is scheduled to be performed on a machine type that has two or more units, the following steps are used.

   a. Rank the parts' operations in the decreasing order of workload.
   b. Assign the first part's operation in the sorted list to a unit of machine, which has the smallest workload assignment so far, in order to distribute the workload among units.
   c. Should there be a tie among the units of a machine type, it is broken in favor of the smallest unit (machine) number.
   d. Should there be a tie among the parts' operations, it is broken in favor of the smallest part number.

e. Remove the part's operation that has just been assigned to a unit of machine from the sorted list.

f. Repeat step b until every part's operation has received its assignment.

Step 1.2: The process of machine grouping is broken into two phases, consisting of identifying the key machines and clustering the remaining machines around the key machines.

Step 1.2.1: Phase I - Identifying the key machines.

The steps associated with phase I are as follows:

(i) Choose the machine k (M_k, $k = 1,\ldots,m$) which has the highest workload from the complete set of m machines (MS). The workload is evaluated as the sum of the processing times contributed by all parts assigned to a machine. Set that machine as the first key machine. In the event of a tie, the machine that processes the maximum number of parts is chosen. Should there be a tie for maximum number of parts, the machine that has the smallest machine number is selected (for example, M_3 is selected because it has a smaller machine number than M_6)

(ii) Form a set of remaining (m-1) machines (RMS)

(iii) From the set RMS, the next key machine is selected as the one that has the highest degree of dissimilarity from the one selected previously. The partial set-covering model proposed by Francis and White (1974) is used for this purpose. First, a functional value for each machine j belonging to the set RMS is evaluated as:

$$Z_j = \sum_{i=1}^{p} \max\left[\left(a_{ij} - a_{ik}\right), 0\right] \forall_j,\ j \neq k$$

where $i = 1,\ldots, p$ parts; $j = 1,\ldots, m$ machines, and $a_{ij} = 1$ if part i has an operation scheduled on machine j, and $a_{ij} = 0$, otherwise. The machine that has the maximum functional value is selected as the next key machine k, as it gives the highest degree of dissimilarity from the one chosen before. In the event of a tie, the machine that has the highest workload is chosen.

(iv) Update the set RMS by deleting the key machine selected in (iii).

(v) Perform phase II (clustering) if the number of key machines selected so far is equal to the number of cells assumed for the problem. Otherwise, repeat step (iii)

Step 1.2.2: Phase II - Clustering the remaining machines around the key machines. The steps associated with phase II are as follows.

(i) From the complete set MS, identify the similarity coefficients for every possible combination of two machines. The similarity coefficient (S_{kl}) between two machines k and l is evaluated as:

$$S_{kl} = \max\left(n_{kl}/n_k, n_{kl}/n_l\right)$$

where,

n_k = number of parts that visit machine k

n_l = number of parts that visit machine l

n_{kl} = number of parts that visit machine k and machine l

(ii) From the table of similarity coefficients created in step (i), find the highest S_{kl} between machines k and l, where $k \notin$ RMS and $l \in$ RMS. Machine k is selected in such a way that it always starts from the first cell (i.e. cell 1) and continues on to the last cell (i.e. the number of cells assumed in the problem instance). If a cell has reached its capacity then the next cell will be used. So, the clustering algorithm will prioritize forming machine groups in the earliest cell, unless the similarity coefficient dictates otherwise.

(iii) Should there be a tie among the similarity coefficients, it is broken in favor of the similarity coefficient that has the most number of parts that visit machine k and machine l or the n_{kl}.

(iv) Assign machine l to the same cell as machine k and remove machine l from the set RMS.

(v) Repeat step (ii) until the set RMS is empty or every machine has received its assignment.

(vi) Assign cell i to location i, where i is the index of cells and it starts from one and continues on to the maximum number of cells assumed in the system.

Step 2: Given the first initial solution (FS_0), which consists of FSm_0 and FSp_0, evaluate its objective function value (Z_0), which is represented by the total service time required to transport all the loads during a planning horizon. At this point, using the information

supplied by step 1, the cell formation problem is ready to be solved. So, with regard to the initial part-machine assignment and the initial machine-location identification, a linear programming model is formulated as a reduced model of the complete cell formation problem developed in Chapter 4. The model is called a reduced model because it only deals with the issues of routing and capacity of the material handling equipment while the issues on part-machine assignment and the machine-location identification have been simplistically resolved. The model formulated derives its form from the binary and general integer non-linear programming model developed in Chapter 4, with the exception that some of the variables from the complete model have been set with specific values. These variables are set with the values according to the initial configuration found in step 1, where the assignment of parts to machines and the placement of machines to their locations are determined using a simple procedure. With these variables known, the complex model from Chapter 4 has been reduced into a simple general-integer linear programming model. The reduction is explained by the elimination of variables S(j,k,p,r) and X(i,p,r) from the complete model. Variable S deals with the assignment of parts to machines, while variable X deals with the assignment of machines to locations. As a result of the elimination, the first four constraints of the complete model can now be removed. In addition, the objective function and the sixth constraint of the complete model are also simplified greatly. The reduced model can be presented as follows:

Minimize

$$Z = \sum_{j=1}^{n} \sum_{k=1}^{k_j+1} \sum_{rt=1}^{rt(r_1,r_2)} F(j,k_j,rt)\, B(j,k_j,rt)$$

subject to

$$\sum_{rt=1}^{rt(r_1,r_2)} B(j,k,rt) = b_j \qquad ;\begin{matrix} j = 1,\ldots,n \\ k = 1,\ldots,k_j+1 \end{matrix} \qquad (1)$$

$$\sum_{j=1}^{n} \sum_{k=1}^{k_j+1} \sum_{rt=1}^{rt(r_1,r_2)} TT(j,k_j,rt)\, B(j,k_j,rt) \le A_g \qquad ;g = 1,\ldots,G \qquad (2)$$

Constraint equation (1) ensures that for each operation of a part, the number of batches transported through different routes is satisfied.

Constraint (2) ensures that feasible capacity is maintained for every material handling equipment used, which in this case is an AGV.

As the above model takes on the form of a general integer-LP problem, it can be easily solved using the commercial integer LP solver (LINDO, 1998). Notice that the above model is only capable of handling feasible solutions. In some circumstances, the infeasible solution could also exist if the capacity of AGVs assumed for the problem instances are too restricted. In this research, both unlimited and limited AGV capacity will be experimented in order to develop a flexible heuristic that can handle variety of needs required by the material handling equipment in the design of CMS. However, in the case of infeasible solutions, the above model would be useless because the solutions obtained from the LP solvers cannot be interpreted. Indeed, those infeasible solutions have no real meaningful value at all. Therefore, a new strategy has to be devised in order to make the reduced model applicable to the infeasible solutions. The new strategy should provide the LP solver with the means of measuring the infeasibility of infeasible solutions. In this research, this new strategy will be called the penalty procedure of the reduced model. It will be integrated into the above reduced model and implemented as follows:

1. In detecting an infeasible solution, the reduced model will add to its objective function value, a constant value (M_1) that is sufficiently large in order to make the infeasible solution become unattractive in comparison to any feasible solutions.
2. In addition, for every unit of AGV capacity that the infeasible solution violated (for example, $U_1 + U_2 + ... + U_G$), the penalty procedure adds another 5 points to its objective function value.

Following the implementation of penalty procedure to the reduced model, the revised model can be presented as follows:

Minimize

$$Z = \left[\sum_{j=1}^{n} \sum_{k=1}^{k_j+1} \sum_{rt=1}^{rt(r_1,r_2)} F(j,k_j,rt)\, B(j,k_j,rt) \right] + M_1 I + 5 * \sum_{g=1}^{G} U_g$$

subject to

$$\sum_{rt=1}^{rt(r_1,r_2)} B(j,k,rt) = b_j \qquad ;j = 1,\ldots,n \;;k = 1,\ldots,k_j+1 \qquad (1)$$

$$\left[\sum_{j=1}^{n} \sum_{k=1}^{k_j+1} \sum_{rt=1}^{rt(r_1,r_2)} TT(j,k_j,rt)\, B(j,k_j,rt) \right] - U_g \le A_g \qquad ;g = 1,\ldots,G \qquad (2)$$

$$\sum_{g=1}^{G} U_g \le M_2 I \qquad (3)$$

where,

- M_1 is a constant that is sufficiently large to differentiate the infeasible solutions from the feasible solutions.
- Variable I is a binary variable that is used to identify an infeasible solution. If the solution is infeasible, I will take on a value of 1, otherwise I will be 0.
- Variable U_g is a real variable that is used to identify the amount of capacity violation for AGV g. If the solution is infeasible, one or more U_g variables will have value greater than zero, otherwise they will all be zero.
- M_2 is a constant that serves as an upper bound on the sum of variable U_g. It is used in conjunction with variable U_g in equation (3) to determine if a solution is infeasible or not.

The revised model now takes on the form of binary and general integer linear programming problem.

Step 3: Given the initial objective function value (Z_0) of the initial solution (FS_0), perform an inside search to explore for a new and better solution. The inside search will only focus on the assignment of parts to machine. It will take the sequence of part-machine assignment associated with the current machine-location identification set by the

outside search and attempt to improve it. At this point, the inside search will take the FSp_0 and use it as the initial parent node to start the search.

Step 4: Using the inside initial solution (FSp_0), generate a set of seeds by perturbing on each operation of a part, but one operation at a time. The perturbation procedure is given by the inside perturbation described at the beginning of this section.

Step 5: Evaluate the objective function value (Z) of each seed using the same procedure outlined in Step 2 of the outside search. From the seed evaluation, select the seed that has the minimum value and use it as the parent node for the subsequent move of the inside tabu search. Perform the inside tabu search to find the optimal/near optimal part-machine assignment corresponding to the current machine-location set by the outside search. The inside tabu search will move from one configuration of part-machine assignment to another and thus, at each move, the parameters that need to be updated are as follows:

(1) Inside Tabu List (ITL).

The ITL is used to prevent the search from cycling and repeating its previous moves. Whenever an inside move is performed, the ITL is updated by admitting the perturbing attributes into the list. The perturbing attributes contain the information on parts, operations of parts and machines that are involved. The perturbing attributes that appear in the ITL indicate that they have been considered at some previous iterations and thus, they receive tabu status. They would not be considered in the next several iterations, unless their tabu status has expired or an aspiration criterion, which allows the tabu status to be overridden, is satisfied. The perturbing attributes will remain tabu for only a certain number of iterations determined by the inside tabu list size. The ITL is updated circularly according to its size. It means that if the ITL was stored up to its size, the oldest entry must be removed before the next entry is stored. Two types of tabu list size are considered in this research; the fixed tabu list size and the variable tabu list size. In determining the formula used for each parameter of the inside tabu search, it is observed that they are closely related to the number of perturbations during the neighborhood search. Therefore, estimation for the number of perturbations performed during the inside neighborhood search is given as follows:

INS = (P*OA*TM) / MT

where,

P = total number of parts in the system

OA = average number of operations per part.

TM = total number of individual machines in the system

MT = total number of machine types in the system.

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:

  Fixed size of ITL

  $= \left\lfloor \sqrt{INS*0.1} \right\rfloor$, if it is a real number with a decimal value < 0.5

  $= \left\lceil \sqrt{INS*0.1} \right\rceil$, if it is a real number with a decimal value ≥ 0.5

- the variable tabu list size for the inside search is determined by the following formula:

  Initial size of ITL

  $= \left\lfloor \sqrt{INS*0.1} \right\rfloor$, if it is a real number with a decimal value < 0.5

  $= \left\lceil \sqrt{INS*0.1} \right\rceil$, if it is a real number with a decimal value ≥ 0.5

  Decreased size of ITL

  $= \left\lfloor \sqrt{INS*0.1}*0.7 \right\rfloor$, if it is a real number with a decimal value < 0.5

  $= \left\lceil \sqrt{INS*0.1}*0.7 \right\rceil$, if it is a real number with a decimal value ≥ 0.5

  Increased size of ITL

  $= \left\lfloor \sqrt{INS*0.1}*1.3 \right\rfloor$, if it is a real number with a decimal value < 0.5

  $= \left\lceil \sqrt{INS*0.1}*1.3 \right\rceil$, if it is a real number with a decimal value ≥ 0.5

(2) Inside Aspiration Criterion (IAC).

The aspiration criterion for the inside search is initially set equal to the inside initial solution, which is the total service time. It is updated with a new total service time, whenever the new total service time evaluated for the current part-machine assignment is found to be better than the total service time found so far.

(3) Inside Candidate List (ICL) and Inside Index List (IIL).

The ICL stores the potential configurations of part-machine assignment that would be used for future perturbations while the IIL stores the configurations that are the local optima of the inside search. The functions and operations of the two lists are described below.

At the start of the inside search, the inside initial solution (FSp_0) is admitted into both ICL and IIL, and used as an initial node for perturbation. The first inside initial solution has its objective function value as (Z_0). When all the seeds of an initial node have been evaluated, the configuration that contributes to the lowest objective function value (Z) is selected and admitted into the ICL and used as the new node for the next perturbation. If the new configuration in ICL has its objective function value (Z_1) smaller than the initial objective function value (Z_0), it would receive a star. 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, which would be admitted into the ICL, is selected as that having the best objective function value (Z_2) from among the seeds perturbed from FSp_1. If $Z_2 \geq Z_1$, 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 while a configuration with double stars is the next local optimum and, therefore, admitted into the IIL. The final solution for the inside search, indicating which part-machine assignment should be used in conjunction with the machine-location identification set by the outside search, is selected as the configuration which gives the best total service time (Z) from among the local optima identified (entries in the IIL).

(4) Stopping Criteria.

To terminate the inside search, two stopping criteria are considered: the number of iterations without improvement (IIWI) and the number of entries into the Inside Index List (EIIL). These two criteria are used together in monitoring the inside tabu search. If one of the criteria is met, the search is terminated.

The IIWI is increased by one if the solution obtained from the current inside move does not show any improvement over the solution of the previous inside move. On the other hand, it is reinitialized back to zero whenever an improvement over the previous inside move is found. The EIIL is increased by one every time an inside move is admitted into the Inside Index List (IIL). The number of entries into the IIL represents the number of local optima found so far during the inside search.

Based on the preliminary experimentation, the stopping criteria is evaluated as follows:

- For the fixed tabu list size, the inside stopping criteria are determined by the formula:

$$\text{IIWI} = \left\lfloor \sqrt[3]{INS*0.4} \right\rfloor, \quad \text{if it is a real number with a decimal value} < 0.5$$

$$= \left\lceil \sqrt[3]{INS*0.4} \right\rceil, \quad \text{if it is a real number with a decimal value} \geq 0.5$$

$$\text{EIIL} = \left\lfloor \sqrt{INS*0.4} \right\rfloor, \quad \text{if it is a real number with a decimal value} < 0.5$$

$$= \left\lceil \sqrt{INS*0.4} \right\rceil, \quad \text{if it is a real number with a decimal value} \geq 0.5$$

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

$$\text{IIWI} = \left\lfloor \sqrt[3]{INS*0.4} * 0.6 \right\rfloor, \text{ if it is a real number with a decimal value} < 0.5$$

$$= \left\lceil \sqrt[3]{INS*0.4} * 0.6 \right\rceil, \text{ if it is a real number with a decimal value} \geq 0.5$$

$$\text{EIIL} = \left\lfloor \sqrt{INS*0.4} \right\rfloor, \quad \text{if it is a real number with a decimal value} < 0.5$$

$$= \left\lceil \sqrt{INS*0.4} \right\rceil, \quad \text{if it is a real number with a decimal value} \geq 0.5$$

The guideline for using IIWI with variable ITL is as follows:

- if there is no improvement within the last IIWI iterations with the initial ITL, then decrease the ITL to the decreased ITL evaluated in step 3.
- if there is no improvement within the last IIWI iterations with the decreased ITL, then increase the ITL to the increased ITL evaluated in step 3.
- if there is no improvement within the last IIWI iterations with the increased ITL, then terminate the inside search.

Step 6: When the inside search is terminated, the optimal/near optimal part-machine assignment would be determined as the one that contributes to the lowest total service time found throughout the inside search. The direction of the search would be returned to the outside search.

Step 7: Perform the outside search, in the same fashion as the inside search, but this time for the machine-location identification. So continuing from step 2, using the outside initial solution (FSm_0) found in step 1 as an initial node, generate a complete set of its seeds by perturbing on each machine occupying a location, with one machine at a time. The perturbation procedure is given by the outside perturbation explained at the beginning of this section.

Step 8: Evaluate the objective function value (Z) of each seed using the same procedure outlined in Step 2. Notice that in the evaluation of each seed, it will always use the same

inside configuration (part-machine assignment) that comes from the result of the inside search for the parent node. In other words, all the seeds of the outside search will utilize the same result obtained from the inside search for their parent node. At each evaluation of the seed, formulate a reduced LP model according to the description given in step 2 of the heuristic algorithm. Based on these evaluations, select the best seed that corresponds to the configuration with the lowest Z value. This seed will be used as the outside configuration of the parent node for the subsequent move of outside search. At this point, a new outside configuration has just been identified but for the inside configuration, it has not changed yet. It is still using the same configuration provided by the parent node. Therefore, step 3 of the heuristic algorithm (inside search) will be executed in order to find a better solution from the current inside configuration. Thus, the best result obtained from the inside search will be used together with the best result obtained from the outside search as the parent node for the subsequent move of the outside search. Notice that the inside search is only executed once for each outside move. It is performed right after the outside search has identified the best outside configuration for the current move. Following that, using the new parent node, step 7 of the heuristic algorithm will be repeated again. The outside search will generate a set of seeds from the new parent node and each seed would need to be evaluated. During the evaluation, the seeds will utilize the same inside configuration provided by the parent node. And, the whole process of step 8 of the heuristic algorithm will repeat itself. Therefore, the outside tabu search, operating as the navigator of the entire search, will keep moving from one parent node to the other parent node until its stopping criteria are met. Depending on the feature of tabu search selected, the long-term memory will also be employed. At the termination of outside search, regardless of the feature used, the optimal/near optimal machine-location configuration provided by the outside search and its best part-machine assignment provided by the inside search would be found.

Step 9: The outside tabu search will move from one configuration of machine-location identification to another and thus, at each move (outside move), its parameters have to be updated. All the parameters used by the outside tabu search is comparable to the one used by the inside tabu search. The only difference is in their association. The outside tabu

search parameters are used for the machine-location identification while the inside tabu search parameters are used for the part-machine assignment. However, for the outside tabu search, the long-term memory is also considered in order to increase its effectiveness in detecting optimal/near optimal solutions. This long-term memory is not considered in the inside search because the machine-location identification has significantly more impact on the cell formation problem than the part-machine assignment. Adding the long-term memory to the inside search will not give substantial benefits to justify the high computational cost and the additional programming effort. However, adding the long-term memory to the outside search might be beneficial because there is wider area to search for the outside search than the inside search. If the outside tabu search relies only on the short-term memory, its capabilities and effectiveness may be limited. Therefore, the parameters of the outside tabu search that also consider the long-term memory are presented below:

(1) Outside Tabu List (OTL).

The OTL, similar to ITL, is used to prevent the search from cycling and repeating its previous moves. Whenever an outside move is performed, the OTL is updated by admitting the perturbing attributes into its list. The perturbing attributes contain the information on machines and locations that are involved. Comparable to the inside tabu search, the parameters used for the outside tabu search are closely related to the number of perturbations performed by the outside neighborhood search (ONS), which is the outside perturbation. The number of perturbations performed by the outside perturbation can also be defined as the number of seeds generated from the parent node by the outside perturbation. It is formulated as follows:

$$\text{ONS} = \left(\sum_{i=1}^{C-1} i \right) * M^2$$

where,

C = total number of cells in the system

M = maximum number of machines assigned to a cell

The summation term basically counts the number of perturbations between any two cells. For example, if a system has only three cells, then the number of

perturbations between any two cells will be 3, which are perturbation between cells 1 and 2, perturbation between cells 1 and 3, and perturbation between cells 2 and 3.

The M^2 term corresponds to the number of times machines are exchanged during a cell perturbation. For example, if M is assigned with 3, then the number of times machines are exchanged during one cell perturbation is 3^2 or 9. Each exchange of machines between any two cells represents one perturbation performed by the outside perturbation.

Based on the preliminary experimentation, the tabu list size for the outside search is evaluated as follows:

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

  Fixed size of OTL

  $= \left\lfloor \sqrt{ONS*0.2} \right\rfloor$, if it is a real number with a decimal value < 0.5

  $= \left\lceil \sqrt{ONS*0.2} \right\rceil$, if it is a real number with a decimal value ≥ 0.5

- the variable tabu list size for the inside search is determined by the following formula:

  Initial size of OTL

  $= \left\lfloor \sqrt{ONS*0.2} \right\rfloor$, if it is a real number with a decimal value < 0.5

  $= \left\lceil \sqrt{ONS*0.2} \right\rceil$, if it is a real number with a decimal value ≥ 0.5

  Decreased size of OTL

  $= \left\lfloor \sqrt{ONS*0.2} * 0.7 \right\rfloor$, if it is a real number with a decimal value < 0.5

  $= \left\lceil \sqrt{ONS*0.2} * 0.7 \right\rceil$, if it is a real number with a decimal value ≥ 0.5

  Increased size of OTL

  $= \left\lfloor \sqrt{ONS*0.2} * 1.3 \right\rfloor$, if it is a real number with a decimal value < 0.5

$$= \left\lceil \sqrt{ONS} * 0.2 * 1.3 \right\rceil \text{ , if it is a real number with a decimal value} \geq 0.5$$

(2) Outside Aspiration Criterion (OAC).

The aspiration criterion for the outside search is initially set equal to the outside initial solution, which is the total service time. It is updated with a new total service time, whenever the new total service time evaluated for the current part-machine assignment is found to be better than the total service time found so far.

(3) Outside Candidate List (OCL) and Outside Index List (OIL).

The OCL stores the potential configurations of machine-location identification that would be used for future perturbations while the OIL stores the configurations that are the local optima of the outside search. The functions and operations of the two lists are similar to the one described earlier for the inside search. The final solution for the outside search, indicating which machine-location identification should be used in conjunction with the part-machine assignment set by the inside search, is selected as the configuration which gives the best total service time (Z) from among the local optima identified (entries in the OIL).

(4) Stopping Criteria.

To terminate the inside search, two stopping criteria are considered: the number of iterations without improvement (OIWI) and the number of entries into the Outside Index List (EOIL). These two criteria are used together in monitoring the outside tabu search. If one of the criteria is satisfied, the search is terminated.

The OIWI is increased by one if a non-improvement solution is found after an outside move is performed and on the other hand, it is reinitialized back to zero whenever an improvement over the previous outside move is found. The EOIL is increased by one every time an outside move is admitted into the Outside Index List (OIL). The number of entries into the OIL represents the number of local optima found so far during the outside search.

Based on the preliminary experimentation, the stopping criteria are evaluated as follows:

- For the fixed tabu list size, the inside stopping criteria are determined by the formula:

$$\text{OIWI} = \left\lfloor \sqrt[3]{INS*0.7} \right\rfloor \quad \text{, if it is a real number with a decimal value} < 0.5$$

$$= \left\lceil \sqrt[3]{INS*0.7} \right\rceil \quad \text{, if it is a real number with a decimal value} \geq 0.5$$

$$\text{OECL} = \left\lfloor \sqrt{ONS*A*0.5} \right\rfloor \quad \text{, if it is a real number with decimal value} < 0.5$$

$$= \left\lceil \sqrt{ONS*A*0.5} \right\rceil \quad \text{, if it is a real number with decimal value} \geq 0.5$$

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

$$\text{OIWI} = \left\lfloor \sqrt[3]{INS*0.7} * 0.6 \right\rfloor \text{, if it is a real number with a decimal value} < 0.5$$

$$= \left\lceil \sqrt[3]{INS*0.7} * 0.6 \right\rceil \text{, if it is a real number with a decimal value} \geq 0.5$$

$$\text{OECL} = \left\lfloor \sqrt{ONS*A*0.5} \right\rfloor \quad \text{, if it is a real number with decimal value} < 0.5$$

$$= \left\lceil \sqrt{ONS*A*0.5} \right\rceil \quad \text{, if it is a real number with decimal value} \geq 0.5$$

The guideline for using OIWI with OTL is as follows:

- if there is no improvement within the last OIWI iterations with the initial OTL, then decrease the OTL to the decreased OTL evaluated in step 3.
- if there is no improvement within the last OIWI iterations with the decreased OTL, then increase the OTL to the increased OTL evaluated in step 3.
- if there is no improvement within the last OIWI iterations with the increased OTL, then terminate the inside search.

where,

A = total number of AGVs in the system

Step 9x: To intensify and diversify the outside search performed in step 9, the advance mechanism of tabu search called the long-term memory, is also employed. The long-term memory for outside search (OLTM) is used to direct the search into a new region that has greater potential of getting superior results. The OLTM can be directed to explore into the area that has provided good solutions previously, for the intensification process or into the area that has received the least attention from previous searches, for the diversification process. The OLTM utilizes a matrix that keeps track the frequency of outside moves attribute. The attribute of interest is the placement of machines at their locations. So, the OLTM matrix keeps record on the number of times that each machine has been assigned to a specific location according to the history of moves obtained by the outside search. The matrix is updated regularly as the outside search progresses. Every time an outside move is performed, the entry in the matrix, which corresponds to the machine-location identification at that point, is increased by one. By keeping track of the frequency of machine-location identification, the OLTM matrix provides the information about which locations have been occupied the most or least frequently by specific machines.

Using the information obtained from the OLTM frequency matrix, a restart configuration is generated. The restarts generate new initial configurations, which are intended to intensify or diversify the search to new regions. The new initial configuration is determined by applying the OLTM frequency matrix to the initial machine-location configuration that was found in step 1. Two types of OLTM are considered in this research: the OLTM based on maximal frequency (OLTM_MAX) and the OLTM based on minimal frequency (OLTM_MIN). The OLTM_MAX is intended to intensify the search by focussing on the area that has been searched frequently in previous searches, while the OLTM_MIN is aimed at diversifying the search by directing the search to the area that has received the least attention in previous searches. The OLTM_MAX generates a restart configuration by fixing several machines to their respective locations according to the maximal entry of the OLTM frequency matrix. When those machines are

fixed to their respective locations, the outside perturbation of tabu search would not perturb on them. This binding of machines to locations will remain throughout the duration of the search for that restart until a new restart is generated again and a new binding other than the previous one will become effective. The OLTM_MIN is implemented in the same way as the OLTM_MAX, except it generates its restart according to the minimal entry of the OLTM frequency matrix. The number of machines that would be fixed to their respective locations is determined by using the formula:

$$FM = \lfloor TM / 8 \rfloor, \text{ if it is a real number with a decimal value} < 0.5$$

$$= \lceil TM / 8 \rceil, \text{ if it is a real number with a decimal value} \geq 0.5$$

where,

TM = total number of machines in the system

Once the restart configuration is obtained, the outside tabu list is reinitialized and the outside search is repeated by performing step 2 and continued on to the last steps until the required number of restarts has been attained. Care should be taken when repeating step 2, instead of starting with the initial configuration found in step 1, make sure that the new restart configuration obtained from step 9x is used. The number of restarts required for the outside search is assumed to be 1, 2, 3 for the small, medium, and large size problem, respectively. At the end of each restart, the OLTM frequency matrix as well as the outside tabu list have to be reinitialized back to zero.

The entire search would be terminated when the required number of restarts for the outside search has been reached. Then, it would return the optimal/near-optimal machine-location configuration together with its optimal/near-optimal part-machine configuration, which are the configurations that give the lowest total service time for the entire search process. Moreover, both configurations will serve as the final solution, either as an optimal or a near optimal solution, for the original research problem.

5.5 Application of the Heuristic to an Example Problem

To illustrate the functionality of the steps associated with the heuristic algorithm, a simple example problem is considered. The example problem covers a CMS that uses a tandem AGV system and a machine-part load matrix. The reason of using tandem AGV system instead of conventional AGV system is presented in Chapter 2. The AGV system will serve the part load movements between four locations including the I/O. It consists of three loops, three AGVs, and three transit points. The load matrix represents the workload generated on machines for processing parts' requirement over the planning horizon. The planning horizon is assumed to be one day. It consists of four parts, five machine types and a maximum of three operations for each part. The layout of the tandem AGV system is presented in Figure 5.2, while the machine-part load matrix is presented in Table 5.1.

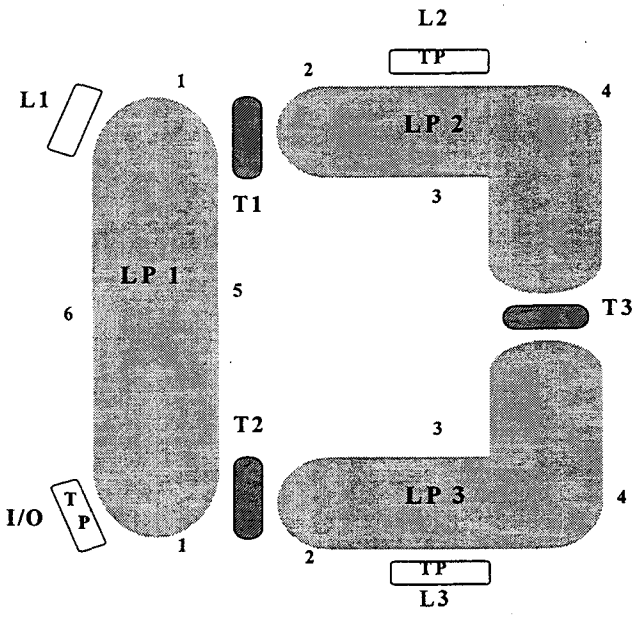

Figure 5.2 Layout of tandem AGV system for the example problem

The layout consists of one input/output (I/O) location and three potential cell locations. The system employs three AGVs and utilizes one AGV for each loop. AGV 1 serves the input/output location and the first location, AGV 2 serves the second location,

and AGV 3 serves the third location. The system has also three transfer points to transfer loads between loops. There are also terminal points assigned for the AGVs at their own loops, which are located at the input/output location, the second location, and the third location represented in Figure 5.2 by AGV 1, 2, and 3, respectively. The terminal point for an AGV is the location where the AGV is held when it is not in use. It is also the place where each AGV receives its order to pick up and deliver loads.

Table 5.1 Machine-part load matrix for the example problem

Machine Type	**P1**	**P2**	**P3**	**P4**	**Total Machine Workload (hrs)**	**Number of Machines**
M1	4			7	11	2
M2		3.5	4, 6		13.5	2
M3	5			3	8	1
M4	6	4.5			10.5	2
M5			5		5	1
Total Part Load	15	8	15	10		
Batch	3	2	3	2		

The machine-part load matrix involves four parts with a maximum of three operations for each part and five machine types. The part routing on each part is assumed to be in the increasing order of machines. For example, part 1 (P1) is processed by machine 1 (M1), M3, and M4, and in that order. A part can also have a non-consecutive operation on the same machine type. This feature has practical implications in the real industry settings. For example, the routing for P3 is assumed in the following order: M2-M5-M2. It implies that M2 is the machine type, which performs the non-consecutive operation on P3. For the purpose of this example, each unit of a machine type is assumed to contribute eight hours of machining time over the planning horizon. A daily planning horizon with eight hours a shift and one shift per day is assumed. Thus, for M1, M2, and M4, the units of machine required for each machine type are 2. The number of batches produced of each part over a planning horizon of 8 hours is also shown in Table 5.1. The

processing times shown in Table 5.1 correspond to producing all of the batches required of each part over an 8 hour planning horizon. In this example, the number of individual machines is limited to nine in order to have a cell structure that requires occupying no more than three machines in each cell since there are only three potential locations.

Also, the following assumptions have been made:

(i) The travel time per unit distance is 5 time units for AGV movement with loads and 3 time units for AGV movement without loads.
(ii) The AGV capacity is considered unlimited for this problem instance.
(iii) The transfer time between loops is considered negligible.
(iv) The cellular movement within a cell is considered negligible
(v) The maximum number of machines that can be assigned to a location is 3.

Step 1: Generate the first initial solution (FS_0) by using the simple procedure reported by Logendran (1991). The FS_0 consists of the first outside initial solution (FSm_0) and the first inside initial solution (FSp_0). The procedure starts by identifying the preliminary part-machine assignment and then finishes off with placing cells to their location.

Step 1.1: The steps associated with the part-machine assignment are as follows:

(i) Operations of parts performed on machine types that have only single unit, will be assigned to those units. For example part 1 operation 2 (P1O2), P3O2, P4O2 are assigned to machine type 3 (M3), M5, and M3, respectively.
(ii) Since M1, M2, and M4 have two units, operations of parts to be performed on them use the following steps:
    a. Operations of parts are arranged in decreasing order of workload for each machine type, as follows:
    M1 : P4O1 (7 hrs) – P1O1 (4 hrs)
    M2 : P3O3 (6 hrs) – P3O1 (4 hrs) – P2O1 (3.5 hrs)
    M4 : P1O1 (6 hrs) – P2O2 (4.5 hrs)
    b. For M1, P4O1 requiring 7 hours is assigned to M11 (the first unit of M1) and P1O1 requiring 4 hours is assigned to M12 (the second unit of M2).

For M2, P3O3 requiring 6 hours is assigned to M21, P3O1 requiring 4 hours is assigned to M22. At this time, the total workload on M21 is 6 hours and M22 is 4 hours. Thus, P2O1 requiring 3.5 hours is assigned to M22.

For M4, P1O1 requiring 6 hours is assigned to M41 and P2O2 requiring 4.5 hours is assigned to M42.

Step 1.2: The steps associated with grouping of machines to form cells are performed in two phases. They are:

Step 1.2.1: Phase I - Identifying the key machines

(i) Select the machine with the highest workload and assign it to the first cell (C1). In this case, M3 has the highest workload (8 hours). So, M3 is selected as the first key machine representing C1.

(ii) Form a set of remaining machines (RMS), which consists of M11, M12, M21, M22, M41, M42, and M5.

(iii)From the set of RMS, the next key machine with the highest degree of dissimilarity from the previous one is selected using the partial set-covering model proposed by Francis and White (1974). The model evaluates the functional value (Z) of each machine belonging to the set RMS. From the evaluation, M22, which has the highest Z value (Z = 2) is selected as the next key machine representing C2.

(iv)Now the set RMS is updated by removing M22 from it.

(v) Since the number of key machines that have been selected is not equal to the number of cells assumed for the problem (3), repeat step (iii). So, the next key machine, which is the third or the final key machine is selected as M11 representing C3. At this point, phase 1 of machine grouping is completed and phase 2, which focuses on clustering the remaining machines around the key machines, begins.

Step1.2.2: Phase II - Clustering the remaining machines around the key machines.

(i) From the complete set of machines, identify the similarity coefficients for every possible combination of two machines. The table of similarity coefficients for this problem instance is presented below.

Table 5.2 Similarity coefficients between any two machines.

Machine	M11	M12	M21	M22	M3	M41	M42	M5
M11	-	0	0	0	1	0	0	0
M12	-	-	0	0	1	1	0	0
M21	-	-	-	1	0	0	0	1
M22	-	-	-	-	0	0	1	1
M3	-	-	-	-	-	1	0	0
M41	-	-	-	-	-	-	0	0
M42	-	-	-	-	-	-	-	0
M5	-	-	-	-	-	-	-	-

(ii) From the set RMS, find the machine that has the highest similarity coefficient to the machine that has been assigned to the cell previously. The highest value for a similarity coefficient between two machines is one. There exist many similarity coefficients with a value of one between two machines. In that case, the algorithm starts from the first machine in the first cell, which is M3. It has a similarity coefficient of value one with M11, M12 and M41. M11 is dropped from consideration because it has been assigned to cell 3. So M12 and M41 are the candidates to be included in cell 1 with M3.

(iii)Since there is a tie between M12 and M41, analyze the number of parts processed by the two similarity coefficients (n_{kl} for M12 and M41). Both similarity coefficients for M12 and M41 process the same number of parts (1). Hence, the smallest unit number of machine would be used, which is M12. M12 is chosen because it has the smaller unit number compared with M41.

(iv)Assign M12 to the same cell as M3 and remove M12 from the set RMS.

(v) Since the set of RMS is not empty yet, repeat step (ii). The following steps show the order of machines that get clustered:

a. With S_{kl} of 1, assign M41 to the same cell as M3, which is cell 1. At this point, cell 1 is full and it has M3, M12, and M41.

b. With S_{kl} of 1, assign M21 to the same cell as M22, which is cell 2.

c. With S_{kl} of 1, assign M42 to the same cell as M22, which is cell 2. At this point, cell 2 is full and it has M22, M21, and M51.

d. With S_{kl} of 0, assign M51 to the same cell as M11, which is cell 3. At this point, the set RMS is empty and the clustering algorithm is terminated. Cell 3 has M11 and M5.

(vi) Once, all the machines have been assigned to their cells, the cells will be placed at their locations according to their indices. For example, cell 1 is assigned to location 1, cell 2 is assigned to location 2, etc. until the number of cells assumed for the problem instance has been reached.

Now, the initial solution (FS_0) can be explained by FSm_0 and FSp_0 as follows:

a. FSm_0 or the initial outside solution will have the following configuration:
[L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5],
which is a sequence of machine-location identification. This sequence describes that location 1 has M3, M12, and M41, location 2 has M22, M21, and M42, and finally location 3 has M11 and M5.

b. FSp_0 or the initial inside solution will have the following configuration:
[P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3],
which is a sequence of part-machine assignment. This sequence describes that operation 1 of part 1 is performed on M12, operation 2 on M3, and operation 3 on M41, operation 1 of part 2 is performed on M22, operation 2 on M42; and etc.

Step 2: Given the two configurations of the initial solution (FS_0), which are the FSm_0 and the FSp_0, formulate a binary and general integer LP model and solve it using a commercial LP solver (LINDO, 1998). The solver yields an objective function value (Z_0) of 1275. The value of Z_0 indicates that the solution is feasible. If the solution was infeasible, the value of Z_0 would be much greater than that. For an infeasible solution, the penalty procedure will add a constant value (M1) that is sufficiently large in order to distinguish the infeasible solution from the feasible solution. In this example problem, M1 is assigned a value of 1500. In addition, for each unit of AGV capacity that is violated, the penalty procedure will add another 5 points. For example, if the LP solver detects that a solution is infeasible, it will immediately add 1500 points to the objective function of the infeasible solution (i.e.1275) and the initial Z_0 will become 2775. In

addition, if the solver indicates that the infeasible solution has exceeded AGV 1 capacity by 100 units and AGV 3 capacity by 50 units, Z_0 will receive additional 5 points for each unit of capacity violated on AGV 1 and AGV 3. Therefore, due to the AGV capacity violation, Z_0 will receive additional 750 points (i.e. 150*5) and the final solution will become 3525.

Step 3: Given Z_0 of 1275, an inside search is performed to explore for a better solution. The outside search will pass the initial part-machine configuration (FSp_0) to the inside search. This FSp_0 derives its initial configuration from the information supplied by step1. The inside search will use the FSp_0 as an initial node to perform inside perturbations. From step 1, the configuration of FSp_0 for this example problem would be [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3].

Step 4: Using the inside initial solution (FSp_0) as a node, generate a set of seeds $Sp(FSp_0)$ by using the inside perturbation. The procedure for inside perturbation is described earlier in Section 5.4. The steps of the procedure as being applied to the example problem are as follows:

(i) Identify the operations of parts in FSp_0 that use machine types that have more than one unit. From the configuration of FSp_0 [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3], create a list of operations of parts that can be perturbed. In this example, part 1 operation 1 (P1O1), which uses machine type 1 that has two units, will be the first to get admitted into the list. Then P1O3, which uses machine type 4 that also has two units, will be admitted next. Following this, P2O1, which uses machine type 2 that has two units, will also join the list. This procedure is continued on until the last operation of the last part is checked for its applicability for inside perturbation. The resulting list will contain P1O1, P1O3, P2O1, P2O2, P3O1, P3O3, and P4O1.

(ii) Starting from the first operation in the list, perform the inside perturbation:

   a. P1O1 will be detached from M12 and attached to M11. Now, M11 will process P1O1 besides P4O1 and its workload will become 11 hours. Since the capacity of

a machine is restricted to 8 hours, M11 is overloaded, meaning that its capacity is violated. The insertion process fails and the swapping process is initiated.

b. P1O1 will swap the machine with another part's operation that uses the same machine type, but a different unit. In this example, P1O1 will swap machine M12 with P4O1 that uses machine M11. After the swap, P1O1 will be processed by M11 and P4O1 will be processed by M12. The capacity of the machines would also have to be checked for feasibility. In this case, there is no capacity violation resulting from the swap. Beside that, the swap process is also restricted to one additional guideline, which enforces the perturbed machines to be located at different locations. If both machines are at the same location, there is no benefit that can result from the perturbation. This guideline is imposed in order to improve the effectiveness of the inside search. In this example, swapping machine between P1O1 and P4O1 is not restricted by the additional guideline because both machines are located separately. The perturbation process will carry on with the next operation in the list.

c. While the insertion or the swapping process is performed on certain operations, all the other operations of parts that are not involved, have to remain at their original assignments.

(iii) The inside perturbation will be performed on every operation of parts that are contained in the list formed in step (i).

(iv) Using the list formed in step (i), each operation of parts is perturbed according to the following order:

a. P1O1 swaps machine with P4O1
b. P1O3 swaps machine with P2O2
c. P2O1 fails to swap machine with P3O3 because both machines are located at the same location
d. P2O2 fails to swap machine with P1O3 because it has been performed before.
e. P3O1 fails to swap machine with P3O3 because both machines are located at the same location.

f. P303 fails to swap machine with P2O1 or P301 because both machines are located at the same location
g. P4O1 fails to swap machine with P1O1 because it has been performed before.

Notice that out of the seven possible perturbations listed in (iv), there are only two perturbations qualified for further evaluation. Therefore, for the inside perturbation, two children are generated out of the parent node (FSp_0).

Step 5: Evaluate the objective function value (Z) of each seed using the same procedure outlined in step 2. Seed 1 of FSp_0 (Sp_1) has Z of 1309 and seed 2 of FSp_0 (Sp_2) has Z of 1465. Since Sp_1 has a better Z than Sp_2 has, Sp_1 is selected as the next parent node of the inside tabu search. Thus, before the inside search continues on perturbing the new parent node, the following parameters need to be updated:

(1) Inside Tabu List (ITL)

As described earlier, ITL is used to prevent the inside search from cycling and repeating its previous moves. Whenever an inside move is performed, ITL is updated by admitting certain attributes into the list. At this point, the first move of inside search has just been performed. The move was performed as the result of swapping P1O1's machine (M12) with P4O1's machine (M11). Therefore, the ITL will take the operation attribute P1O1 together with the machine attribute M12 and store them as the first entry in its list. This means that P1O1 has previously used M12 in the most recent iteration and it has just been changed to process on another machine (M11 in this case). So, for the next several iterations or inside moves depending on the ITL size, P1O1 should not be perturbed back to M12. In other words, P1O1 should use machines other than M12 as long as it remains tabu in the ITL. As for the size ITL, there are two types of tabu list size used, the fixed ITL and the variable ITL. The formulae used in determining both types are given in Section 5.4. As mentioned earlier in Section 5.4, the parameters used for the inside tabu search will be closely related to the number of perturbations performed during its neighborhood search, which is the inside perturbation. Therefore, this number is estimated first as:

INS = (P*OA*TM) / MT = (4*2.5*8)/5 = 16.

where,

P = total number of parts in the system

OA = average number of operations per part

TM = total number of individual machines in the system

MT = total number of machine types in the system.

Proceeding with the list size of ITL, for each type of ITL, it is evaluated as follows:

- For fixed ITL = $\sqrt{INS * 0.1} = \sqrt{16 * 0.1} = 1.27$ or 1 as it is rounded down to its closest integer.
- For variable ITL, there will be three sizes,
  - The initial size = $\sqrt{INS * 0.1} = \sqrt{16 * 0.1} = 1.27$ or 1 as it is rounded down to its closest integer
  - The decreased size = $\sqrt{INS * 0.1} * 0.7 = \sqrt{16 * 0.1} * 0.7 = .89$ or 1 as it is rounded up to its closest integer.
  - The increased size = $\sqrt{INS * 0.1} * 1.3 = \sqrt{16 * 0.1} * 1.3 = 1.64$ or 2 as it is rounded up to its closest integer.

(2) Inside Aspiration Criterion (IAC).

The IAC for the inside search is initially set equal to the objective function value of the inside initial solution (Z_0), which is 1275. So, in_AL is set to 1275 and it is updated when a smaller total service time is found throughout the inside search.

(3) Inside Candidate List (ICL) and Inside Index List (IIL)

At the start of inside search, the initial part-machine assignment configuration (FSp_0) is admitted into both ICL and IIL. If a new configuration from an inside move is found to be better than FSp_0, then it will receive a star, to indicate that it has the potential of becoming the next local optimum. At this point of the example problem, the configuration of the first move will be admitted into the ICL. Since the Z value of

the first move is not better than the Z_0 value, it would not receive a star. So, the entries into the ICL and IIL are as follows:

- ICL = { [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3],
  [P1:M11,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M12,M3] }
- IIL = { [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3] }

(4) Stopping Criteria

To terminate the inside search, the number of iterations without improvement (IIWI) and the number of entries into the Inside Candidate List (IECL) are used simultaneously. The IIWI is increased by one every time a non-improvement move is made and reset to zero if an improvement over the previous inside move is found. The EIIL is increased by one every time an inside move is admitted into the IIL. The formula used in the application of IIWI and EIIL, as being applied to the example problem, is as follows:

- For fixed ITL,

  IIWI = $\sqrt[3]{INS * 0.4} = \sqrt[3]{16 * 0.4} = 1.86$ or 2 as it is rounded down to its closest integer.

  EIIL = $\sqrt{INS * 0.4} = \sqrt{16 * 0.4} = 2.53$ or 3 as it is rounded up to its closest integer.
- For variable ITL,

  IIWI = $\sqrt[3]{INS * 0.4} * 0.6 = \sqrt[3]{16 * 0.4} * 0.6 = 1.04$ or 1 as it is rounded down to its closest integer.

  EIIL = $\sqrt{INS * 0.4} = \sqrt{16 * 0.4} = 2.53$ or 3 as it is rounded up to its closest integer.

  The guideline for using IIWI with variable ITL is as follows:
  - if there is no improvement within the last IIWI (1) iteration with the initial ITL, then decrease the ITL to the decreased ITL evaluated in step (1).
  - if there is no improvement within the last IIWI (1) iteration with the decreased ITL, then increase the ITL to the increased ITL evaluated in step (1).
  - if there is no improvement within the last IIWI (1) iteration with the increased ITL, then terminate the inside search.

During the course of inside search, either for fixed or variable ITL, the search will be terminated if one of the stopping criteria, the IIWI or the EIIL, is met. At this point of the example problem, both stopping criteria are not satisfied yet because so far there is only one non-improvement move (IIWI=1) and one entry into the IIL (EIIL=1).

Continuing with the next move of the inside search (i.e. the second move), the following parent node is used for inside perturbation [P1:M11,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M12,M3]. Out of this parent node, there is only one child that could be generated, which is the result of swapping P1O3's machine with P2O2's machine. This seed will have a configuration of [P1:M11,M3,M42; P2:M22,M41; P3:M22,M5,M21; P4:M12,M3] and a Z value of 1499. In addition, the following parameters of the inside tabu search are also updated:

(1) ITL will now have the new operation attribute P1O3 and the new machine attribute M41 in its list. The older entry is removed from the list since the size of ITL is only one.

(2) In_AL will still have the Z_0 value 1275 since the Z value of the new move (1499) is greater than its current in_AL (1275).

(3) ICL will now have three entries in its list, while IIL still has only one entry in its list.

- ICL = { [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3],
  [P1:M11,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M12,M3],
  [P1:M11,M3,M42; P2:M22,M41; P3:M22,M5,M21; P4:M12,M3] }
- IIL = { [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3] }

(4) IIWI is increased by one and becomes two. EIIL does not change and still remains one.

At this stage, one of the stopping criteria (IIWI) is met and the inside tabu search will be terminated. The inside search will return the best configuration found so far to the outside search. Since there is no improvement that can be made on the initial node (FSp_0), the inside search will return FSp_0 back to the outside search. So, the best result obtained from the initial machine location configuration (FSm_0) = [L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5] and the initial part machine configuration (FSp_0) = [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3] is the same FSp_0.

Step 6: The inside search is terminated after two iterations with FSp_0 as its best part-machine assignment configuration. The direction of the search is returned to the outside search and the result of the inside search will be used as the part-machine assignment for the seeds generated by the outside perturbation from the current parent node.

Step 7: Perform the outside search, in the same fashion as the inside search, but for identifying the location of machines. So continuing from step 2, using the initial outside solution (FSm_0) = [L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5], found in step 1 as an initial node, a set of its seeds $Sm(FSm_0)$ is generated by using the outside perturbation. The procedural steps for the outside perturbation, as being applied to this example problem, are as follows:

(i) Perturb on each machine occupying a location, with one machine at a time.

(ii) First, take the first machine from the first location, which is M3 and insert it to the second location. If the second location is full, the insertion process fails and the swapping process is initiated.

(iii) Swap the first machine from the first location (M3) with the first machine from the second location (M22). If the swapping process is successful, a new machine location configuration is admitted into the list of seeds or $Sm(FSm_0)$.

(iv) Still using M3, repeat step (ii) and (iii) for each machine located separately from M3. So, M3 will swap places with M21, M42, M11, and M5, in succession and it will also get inserted into location 3 because location 3 is not full yet.

(v) Once finished with M3, M12 will be perturbed in the same manner as M3. So using M2, repeat step (ii) and (iii) with M22, M21, M42, M11, and M5. The same process is also repeated for M41. Once all the machines in the first location have been perturbed, the machines in the second location will be perturbed next. The perturbation is performed using the same procedure applied for the machines in the first location. The only difference is that the machines in the second location will not be perturbed again with the machines in the first location. The perturbation only works in one direction, which is to the right of the location sequences.

(vi) This machine-location (outside) perturbation is also called the neighborhood search of the outside tabu search. When this outside perturbation performed on the node FSm_0 is completed, its seeds will consist of 27 new machine location configurations. These configurations can be summarized as the perturbation performed on every unique combination of two different locations and for each machine contained in each of the two different locations. The list of 27 new configurations, formed by sequentially following the description above, is as follows:

Starting from FSm_0 = [L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5]

- perturbation between location 1 and 2
  - using M3 as the subject of perturbation
    - Sm1 = [L1:M22, M12, M41; L2: M3, M21, M42; L3:M11, M5]
    - Sm2 = [L1:M21, M12, M41; L2: M22, M3, M42; L3:M11, M5]
    - Sm3 = [L1:M42, M12, M41; L2: M22, M21, M3; L3:M11, M5]
  - using M12 as the subject of perturbation
    - Sm4 = [L1:M3, M22, M41; L2: M12, M21, M42; L3:M11, M5]
    - Sm5 = [L1:M3, M21, M41; L2: M22, M12, M42; L3:M11, M5]
    - Sm6 = [L1:M3, M42, M41; L2: M22, M21, M12; L3:M11, M5]
  - using M41 as the subject of perturbation
    - Sm7 = [L1:M3, M12, M22; L2: M41, M21, M42; L3:M11, M5]
    - Sm8 = [L1:M3, M12, M21; L2: M22, M41, M42; L3:M11, M5]
    - Sm9 = [L1:M3, M12, M42; L2: M22, M21, M41; L3:M11, M5]
- perturbation between location 1 and 3
  - using M3 as the subject of perturbation
    - Sm10 = [L1:M12, M41; L2: M22, M21, M42; L3:M11, M5, M3]
    - Sm11 = [L1:M11, M12, M41; L2: M22, M21, M42; L3:M3, M5]
    - Sm12 = [L1:M5, M12, M41; L2: M22, M21, M42; L3:M11, M3]
  - using M12 as the subject of perturbation
    - Sm13 = [L1:M3, M41; L2: M22, M21, M42; L3:M11, M5, M12]
    - Sm14 = [L1:M3, M11, M41; L2: M22, M21, M42; L3:M12, M5]
    - Sm15 = [L1:M3, M5, M41; L2: M22, M21, M42; L3:M11, M12]
  - using M41 as the subject of perturbation

- Sm16 = [L1:M3, M12; L2: M22, M21, M42; L3:M11, M5, M41]
- Sm17 = [L1:M3, M12, M11; L2: M22, M21, M42; L3:M41, M5]
- Sm18 = [L1:M3, M12, M5; L2: M22, M21, M42; L3:M11, M41]

- perturbation between location 2 and 3
  - using M22 as the subject of perturbation
    - Sm19 = [L1:M3, M12, M41; L2: M21, M42; L3:M11, M5, M22]
    - Sm20 = [L1:M3, M12, M41; L2: M11, M21, M42; L3:M22, M5]
    - Sm21 = [L1:M3, M12, M41; L2: M5, M21, M42; L3:M11, M22]
  - using M21 as the subject of perturbation
    - Sm22 = [L1:M3, M12, M41; L2: M22, M42; L3:M11, M5, M21]
    - Sm23 = [L1:M3, M12, M41; L2: M22, M11, M42; L3:M21, M5]
    - Sm24 = [L1:M3, M12, M41; L2: M22, M5, M42; L3:M11, M21]
  - using M42 as the subject of perturbation
    - Sm25 = [L1:M3, M12, M41; L2: M22, M21; L3:M11, M5, M42]
    - Sm26 = [L1:M3, M12, M41; L2: M22, M21, M11; L3:M42, M5]
    - Sm27 = [L1:M3, M12, M41; L2: M22, M21, M5; L3:M11, M42]

Step 8: Using the procedure outlined in step 2, evaluate the objective function value (Z) for each seed. Notice that in each evaluation, the result of the inside search, which is the best part-machine assignment for the parent node (FSp_0), is used together with the machine-location configuration of each seed. The values so obtained are presented below:

$Z_1 = 1557$	$Z_{10} = 1596$	$Z_{19} = 1128$
$Z_2 = 1475$	$Z_{11} = 1354$	$Z_{20} = 1098$
$Z_3 = 1575$	$Z_{12} = 1450$	$Z_{21} = 1098$
$Z_4 = 1456$	$Z_{13} = 1309$	$Z_{22} = 1038$
$Z_5 = 1374$	$Z_{14} = 1281$	$Z_{23} = 1008$
$Z_6 = 1474$	$Z_{15} = 1377$	$Z_{24} = 1008$
$Z_7 = 1447$	$Z_{16} = 1309$	$Z_{25} = 1335$
$Z_8 = 1365$	$Z_{17} = 1281$	$Z_{26} = 993$
$Z_9 = 1465$	$Z_{18} = 1377$	$Z_{27} = 1305$

Out of the 27 configurations, configuration 26 has the lowest objective function value (993) and it is identified as the best seed. So, seed number 26 is selected as the outside configuration for the next parent node of the outside search. At this point, though a new outside configuration has been identified, the inside configuration is still the same as that used by the parent node. Therefore step 3 or the inside search will be executed in order to find a better solution from the current inside configuration. Since the inside search is not able to improve on the current solution (993), the same inside configuration as that used by the parent node (FSp_0) will be used with the next parent node of the outside search. Thus, the new parent node of the outside search will consist of the new outside configuration (Nm_{26} of FSm_0) and the former inside configuration (Np_1 of FSp_0). For the first move of the outside search, the best solution is found with machine-location configuration of [L1:M3, M12, M41; L2: M22, M21, M11; L3:M42, M5] and part-machine configuration of [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3]. The best solution has value of 993 and it will be used as the next parent node of the outside search. The new parent node (FS_1) consists of the new outside configuration (FSm_1) and the new inside configuration (FSp_1). With FS_1, step 7 is repeated in its entirety until the stopping criteria of the outside are met.

Step 9: At each move of the outside search, the following parameters of outside tabu search have to be updated:

(1) Outside Tabu List (OTL)

Whenever an outside move is performed, the OTL is updated by admitting the perturbing attributes into its list. In this example, for the first outside move, the first entry into the OTL list will contain two pairs of attributes, which are L2-M42 and L3-M11. It means that M42 has previously resided at location 2 and M11 has previously resided at location 3. So, for the next several iterations or outside moves depending on the OTL size, M42 should not be perturbed back to location 2 and M11 should not be perturbed back to location 3. These attributes are rendered tabu by the outside search. As for the size of OTL, two types of tabu list size are used: the fixed OTL and the variable OTL. The formulae used in determining both types are given in Section 5.4. Comparable to the inside search, the parameters used for the outside search will

be closely related to the number of perturbations performed by the outside neighborhood search, which is the outside perturbation. The formula used to calculate the number of perturbations performed by the outside perturbation is also given in Section 5.4. It is evaluated as follows:

$$\text{ONS} = \left(\sum_{i=1}^{C-1} i\right) * M^2 = \left(\sum_{i=1}^{3-1} i\right) * 3^2 = (1+2)*9 = 27 \text{ perturbations}$$

where,

C = total number of cells in the system

M = maximum number of machines assigned to a cell

Proceeding with the size of OTL, each type of OTL is evaluated as follows:

- For fixed OTL $= \sqrt{\text{ONS} * 0.2} = \sqrt{27 * 0.2} = 2.32$ or 2 as it is rounded down to its closest integer.
- For variable OTL, there will be three sizes,
  - The initial size $= \sqrt{\text{ONS} * 0.2} = \sqrt{27 * 0.2} = 2.32$ or 2 as it is rounded down to its closest integer.
  - The decreased size $= \sqrt{\text{ONS} * 0.2} * 0.7 = \sqrt{27 * 0.2} * 0.7 = 1.63$ or 2 as it is rounded up to its closest integer.
  - The increased size $= \sqrt{\text{ONS} * 0.2} * 1.3 = \sqrt{27 * 0.2} * 1.3 = 3.02$ or 3 as it is rounded down to its closest integer.

(2) Outside Aspiration Criterion (OAC)

The OAC for the outside search is initially set equal to the objective function value of the outside initial solution (Z_0), which is 1275. So, out_AL is set to 1275 and it is updated, whenever a smaller total service time is found during the outside search.

(3) Outside Candidate List (OCL) and Outside Index List (OIL)

At the start of outside search, the initial machine-location identification configuration (FSm_0) is admitted into both OCL and OIL. If a new configuration from an outside move is found to be better than FSm_0, then it will receive a star, to indicate

that it has the potential of becoming the next local optimum. In this example problem, the first entry into OCL and OIL is FSm_0, with a value of 1275. The second entry into OCL is FSm_1, which is seed 26 from FSm_0, with a value of 993. Since FSm_1 has a better total service time than FSm_0, it would receive a star to indicate that it has the potential of becoming the next local optimum. So, the entries into the OCL and OIL at the present time are:

- OCL = { [L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5],
  [L1:M3, M12, M41; L2: M22, M21, M11; L3:M42, M5]* }
- OIL = { [L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5] }

(4) Stopping Criteria

To terminate the outside search, the number of iterations without improvement (OIWI) and the number of entries into the Outside Index List (EOIL) are simultaneously employed. The OIWI is increased by one every time a non-improvement move is made and reset to zero if an improvement over the previous outside move is found. The EOIL is increased by one every time an outside move is admitted into the OIL. The formula used in the application of OIWI and EOIL, as being applied to the example problem, are:

- For fixed OTL,

  $OIWI = \sqrt[3]{INS * 0.7} = \sqrt[3]{27 * 0.7} = 2.66$ or 3 as it is rounded up to its closest integer.

  $OECL = \sqrt{ONS * A * 0.5} = \sqrt{27 * 3 * 0.5} = 6.36$ or 6 as it is rounded down to its closest integer.

- For variable OTL,

  $OIWI = \sqrt[3]{INS * 0.7} * 0.6 = \sqrt[3]{27 * 0.7} * 0.6 = 1.60$ or 2 as it is rounded up to the closest integer.

  $OECL = \sqrt{ONS * A * 0.5} = \sqrt{27 * 3 * 0.5} = 6.36$ or 6 as it is rounded down to its closest integer.

  where,

  A = total number of AGVs in the system

The guideline for using OIWI with variable OTL is as follow:

- if there is no improvement within the last OIWI (2) iteration with the initial OTL, then decrease the OTL to the decreased OTL evaluated in step (1).
- if there is no improvement within the last OIWI (2) iteration with the decreased OTL, then increase the OTL to the increased OTL evaluated in step (1).
- if there is no improvement within the last OIWI (2) iteration with the increased OTL, then terminate the inside search.

During the course of outside search, either for fixed or variable OTL, the search will be terminated if one of the stopping criteria, OIWI or EOIL is met. In this example problem, where the first outside move has just been performed, the status of OIWI is 0 because there is an improvement in the solution of the first move (993) over the initial solution (1275). Moreover, the status of EOIL will still be 1. The results for the outside search with fixed tabu list size, short-term memory, and starting with FS_0 as the very initial solution, are shown in Table 5.3. Notice that FS_0 consists of FSm_0 = [L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5] and FSp_0 = [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3]. Even though at each move of outside search, a new outside and inside configurations are identified, only the outside configuration will be shown in Table 5.3. The inside configuration is purposely left out just to make the presentation looks more compact.

The outside search is terminated after 15 iterations because one of the stopping criteria for the outside search has been met. In this particular case, the EOIL has reached its maximum, which is 6 (for fixed tabu list size). The best configuration for the outside search is found at iteration 6 with value of 861.

Table 5.3 Results obtained for the outside search starting with FS_0

Iteration No.	Entry into OCL	Total Service Time	Entry into OIL
1	[{M3,M12,M41} {M22,M21,M42} {M11,M5}]**	1275	Yes
2	[{M3,M12,M41} {M22,M21,M5} {M11,M42}]*	993	
3	[{M3,M12,M41} {M21,M5} {M11,M42,M22}]**	920	Yes
4	[{M3,M11,M41} {M21,M5} {M12,M42,M22}]	920	
5	[{M3,M11,M41} {M21,M12} {M5,M42,M22}]	950	
6	[{M3,M11,M41} {M42,M12} {M5,M21,M22}]**	861	Yes
7	[{M3,M11,M12} {M42,M41} {M5,M21,M22}]	901	
8	[{M42,M11,M12} {M3,M41} {M5,M21,M22}]	961	
9	[{M42,M12} {M11,M3,M41} {M5,M21,M22}]**	892	Yes
10	[{M42,M41} {M11,M3,M12} {M5,M21,M22}]	924	
11	[{M21,M41} {M11,M3,M12} {M5,M42,M22}]	997	
12	[{M21,M5} {M11,M3,M12} {M41,M42,M22}]**	964	Yes
13	[{M21,M5} {M11,M3,M41} {M12,M42,M22}]	973	
14	[{M21,M12,M5} {M11,M3,M41} {M42,M22}]**	965	Yes
15	[{M21,M12,M5} {M3,M41} {M11,M42,M22}]	1042	

It has machine-location configuration of [L1:M3,M11,M41; L2:M42,M12; L3:M5,M21,M22] and part-machine configuration of [P1:M11,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M12,M3]. The outside candidate list (OCL) has 15 entries in its list, while the outside index list (OIL) has 6 entries in its list. At this point of termination, the short-term memory of the outside search has just been completed and a decision has to be made whether the search should proceed with the long-term memory or not.

Step 9x: To intensify and diversify the outside search, the long-term memory is implemented. The outside long-term memory (OLTM) utilizes a matrix that keeps track

the frequency of outside moves attribute during the course of outside search. The attribute of interest is the placement of machines at their locations. Every time an outside move is performed, the entry in the matrix that corresponds to the machine-location identification at that point is increased by one. Originally, the entries in OLTM are all initialized to zero. At the start of outside search, when an initial machine-location configuration has just been identified with FSm_0 = [L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5] (refer to step 1), the OLTM matrix will have the following entries, as presented in Table 5.4

Table 5.4 Entries into the OLTM matrix as FSm_0 has just been identified.

Machine ID	Location 1	Location 2	Location 3
M11	0	0	1
M12	1	0	0
M21	0	1	0
M22	0	1	0
M3	1	0	0
M41	1	0	0
M42	0	1	0
M5	0	0	1

As the outside search progresses, the OLTM matrix is updated regularly. The corresponding OTLM frequency matrix for the outside search when it is terminated is presented in Table 5.5. It describes that there are all together 15 outside moves that have just been performed including the initial solution. The information from the above matrix can be interpreted as out of the 15 outside moves, 5 times M11 resides at location 1, 6 times at location 2, and 4 times at location 3. Similarly, out of the 27 outside moves, 2 times M22 resides at location 2, 13 times at location 3, and none of them at location 1. Similar explanations can be provided for other machines as well.

Table 5.5 Entries into the OLTM matrix at the time of termination.

Machine ID	Location 1	Location 2	Location 3
M11	5	6	4
M12	8	5	2
M21	5	5	5
M22	0	2	13
M3	7	8	0
M41	8	6	1
M42	3	3	9
M5	4	3	8

Using the information obtained from the OLTM frequency matrix, a restart configuration is generated. There are two types of restarts considered in this research, the restart that is based on maximal frequency (OLTM_MAX) and the restart that is based on minimal frequency (OLTM_MIN). The OLTM_MAX, which is intended to intensify the search, fixes several machines to their respective locations according to the maximal entry into the frequency matrix. On the other hand, the OLTM_MIN, which is intended to diversify the search, fixes several machines to their respective locations according to the minimal entry into the frequency matrix. In this example problem, the number of machines that would be fixed to their respective locations (FM), according to the formula described in Section 5.4, is:

$FM = TM/8 = 8/8 = 1$

where,

TM = total number of machines in the system

So, if the OLTM_MAX is used with the example problem, the maximal entry into the frequency matrix is found to be 13, and it corresponds to M22 at location 3. In case of a tie for maximal entry, the row wise first best strategy is used to break the ties. Therefore, based on the OLTM_MAX, the new initial configuration for the first restart will be constructed by fixing M22 to location 3, and rearranging the initial machine-location

configuration (FSm_0). If the initial machine-location configuration (FSm_0) is [L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5], then the new restart configuration would be [L1:M3, M12, M41; L2: M21, M42; L3:M11, M5, M22], where M22 has now been placed at location 3. Here, M22 is underlined to indicate that M22 is now fixed at location 3 and throughout the rest of the search with the first restart. Notice that when placing M22 at location 3, there is no machine that need to be displaced from location 3. The reason is that location 3 still has enough capacity to accept an additional machine. However, if location 3 is full (i.e. location 3 has already 3 machines), one of the machines has to be moved out of location 3 and assigned to the location where M22 is originated, which is location 2 in this case. To pick the machine that would leave location 3 and go to location 2, a simple procedure is used. First, identify the machine in location 3 that has the most frequency residing in location 2. For the sake of describing this scenario, suppose that M11 or M5 has to be moved out of location 3 and assigned to location 2. From the frequency matrix table (refer to Table 5.5), M11 has 6 entries associated with location 2 and M5 has 3 entries associated with location 2. Since M11 has more entries into location 2 than M5, M11 will be selected. This procedure basically uses the history of previous moves to decide which machine has the most tendency to reside at a certain location based on the maximal frequency strategy. For minimal frequency strategy, the rules will always pick on the machine that has the least tendency to reside at a certain location.

Once a restart seed is obtained, the search for the next restart would be performed in a similar fashion as the early search when an initial solution has just been identified except the machine that receives a "fix" identification (underline) would not be perturbed. So, starting from Step 3 of the heuristic algorithm, an inside search would be performed on the first restart seed. Then, the process will continue on with the next step of the heuristic algorithm until the search for the first restart seed is terminated. Based on the OLTM_MAX, the results obtained with the first long-term memory restart is shown in Table 5.6

Table 5.6 Outside search results for the first restart based on maximal frequency

Iteration No.	Entry into OCL	Total Service Time	Entry into OIL
1	[{M3,M12,M41} {M21,M42} {M11,M5,M22}]**	1098	Yes
2	[{M3,M12,M41} {M11,M42} {M21,M5,M22}]**	861	Yes
3	[{M3,M12,M11} {M41,M42} {M21,M5,M22}]	901	
4	[{M42,M12,M11} {M41,M3} {M21,M5,M22}]	961	
5	[{M42,M11} {M12,M41,M3} {M21,M5,M22}]**	892	Yes
6	[{M42,M41} {M12,M11,M3} {M21,M5,M22}]	924	
7	[{M21,M41} {M12,M11,M3} {M42,M5,M22}]	997	
8	[{M21,M5} {M12,M11,M3} {M42,M41,M22}]**	964	Yes
9	[{M21,M5} {M12,M41,M3} {M42,M11,M22}]	973	
10	[{M21,M11,M5} {M12,M41,M3} {M42,M22}]**	965	Yes
11	[{M21,M11,M5} {M41,M3} {M42,M12,M22}]	1042	
12	[{M21,M11,M12} {M41,M3} {M42,M5,M22}]*	1034	
13	[{M21,M12} {M11,M41,M3} {M42,M5,M22}]**	965	Yes
14	[{M41,M12} {M11,M21,M3} {M42,M5,M22}]	1091	

The outside search for the first restart is terminated after 14 iterations because the number of entries into the Outside Index List (EOIL) has reached its maximum, which is 6 (for fixed tabu list size). The whole search is also terminated because the number of restarts for the outside search has been reached (1). The best configuration from the first restart of the outside search is found at the second iteration with value of 861. It has machine-location configuration of [L1:M3,M12,M41; L2:M11,M42; L3:M21,M5,M22] and part-machine configuration of [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3]. Table 5.7 summarizes the best solution found at the initial and the first restart of the search.

Table 5.7 Summary of results for the entire search process

Restart No.	Configuration of the best solution	Total Service Time
Initial	FSm_6: [L1:M3,M12,M41; L2:M42,M11; L3:M5,M21,M22] FSp_6: [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3]	861
First	FSm_2: [L1:M3,M12,M41; L2:M11,M42; L3:M21,M5,M22] FSp_2: [P1:M12,M3,M41; P2:M22,M42; P3:M22,M5,M21; P4:M11,M3]	861

The above table shows that the best solution for both short-term and long-term memory of the outside search has the same results. This could possibly occur due to two reasons. First, both solutions found are optimal, which is not true in this case as seen in Chapter 6. Second, for this problem instance, the OLTM_MAX is not very effective in directing the search to a new region, which is truly the case here. Even though the long-term memory based on maximal frequency was not able to identify a better solution for this example problem, there is still another strategy based on minimal frequency that could be used. Indeed, when the OLTM_MIN is used, the outside search is able to identify the optimal solution for this example problem, which is 831. This result is proven optimal as presented later in Chapter 6.

If the OLTM_MIN is used in this example problem, the minimal entry into the frequency matrix has to be identified. In this case, it is found to be zero, which corresponds to both M22 at location 1 and M3 at location 3. Since the row wise first best strategy is used to break the ties, M22 is selected. If the initial machine-location configuration (FSm_0) is [L1:M3, M12, M41; L2: M22, M21, M42; L3:M11, M5], then the new restart configuration would be [L1:M3,M22,M41; L2:M12,M21,M42, L3:M11,M5], where M22 has now been placed at location 1. Here, M22 is underlined to indicate that M22 is now fixed at location 1 and throughout the rest of the search with the first restart. Different from the situation that occurred in OTLM_MAX, where no machine need to be displaced when fixing a machine to a location, in OLTM_MIN, a machine has to be moved out of location 1 and assigned to location 2 when attempting to place M22 to location 1. Checking with the table of frequency matrix (refer to Table 5.5),

M12 is selected because it has the least frequency to reside in location 2 in comparison to M3 and M41. Notice that the least frequency is used here for OLTM_MIN in contrary to the most frequency usage for OLTM_MAX. This restart configuration has an initial Z value of 1441. Performing the search in a similar fashion as the OTLM_MAX, the results for the OLTM_MIN are presented in Table 5.8. The whole search is terminated after 13 iterations. Unlike the previous searches, in which they are terminated by the EOIL, this search is terminated because the number of iterations without improvement (OIWI) has reached its maximum, which is 3 in this example problem. It has 13 entries into OCL and 3 entries into OIL. The best configuration is found at the sixth iteration with a value of 831. This value is also the optimal solution for this problem instance based on the unlimited case of AGV capacity. Its optimality is proven in the next chapter together with the solution for several other problem instances.

Table 5.8 Outside search results for the first restart based on minimal frequency

Iteration No.	**Entry into OCL**	**Total Service Time**	**Entry into OIL**
1	[{M3,M22,M41} {M12,M21,M42} {M11,M5} **	1441	Yes
2	[{M3,M22,M41} {M12,M42} {M11,M5,M21} *	1174	
3	[{M5,M22,M41} {M12,M42} {M11,M3,M21} *	1108	
4	[{M5,M22,M41} {M12,M11} {M42,M3,M21} *	1072	
5	[{M5,M22,M41} {M21,M11} {M42,M3,M12} *	877	
6	[{M5,M22,M21} {M41,M11} {M42,M3,M12} **	831	Yes
7	[{M5,M22,M21} {M41,M42} {M11,M3,M12}	944	
8	[{M5,M22,M21} {M3,M42} {M11,M41,M12}	1098	
9	[{M5,M22,M21} {M3,M12,M42} {M11,M41} *	1017	
10	[{M5,M22,M21} {M3,M12,M11} {M42,M41} **	1008	Yes
11	[{M5,M22,M41} {M3,M12,M11} M42,M21}	1042	
12	[{M22,M41} {M3,M12,M11} {M42,M21,M5}	1042	
13	[{M22,M41} {M3,M12,M42} {M11,M21,M5}	1051	

6. OPTIMALITY OF TABU SEARCH-BASED HEURISTIC

6.1 Determination of the Optimal Solution

This chapter describes the procedure used for determining the optimal solutions of small problems used with the tabu search-based heuristic. It is used as a mechanism to assess the quality of the solution determined from the heuristic algorithm developed in Chapter 5. To determine the quality of a heuristic, the solution produced by the heuristic has to be compared to the global optimal solution of the same problem. In this research, the original problem could be solved optimally by formulating it as a mathematical programming model. However, as mentioned in Chapter 4, the mathematical model for the problem investigated here belongs to the NP-hard class. Therefore, even for a small problem, computationally it would be very difficult to find its optimal solution. Despite the difficulty, an attempt was still made to determine the global optimal solution of the original problem so that it could be compared with the solutions obtained from the heuristic. Indeed, with the aid of several transformation techniques, a powerful linear solver, and a specially coded algorithm, the optimal solution for a small problem can be determined in this research. The transformation techniques are used to transform the original polynomial programming model into a pure binary-integer linear programming model. The linear solver is used to solve the transformed model to obtain its optimal solution. The specially coded algorithm is used to decompose the original problem into smaller sub problems. It is also used to coordinate the transformation of the original model and the execution of the linear solver systematically. In the following paragraphs, the functionality of each component used in this procedure is described in more detail.

6.1.1 Construction of Small Problem Structure

The procedure starts by identifying a sufficiently small linear programming model that can be solved by a commercial linear solver effectively. The commercial linear solver employed here is Hyper Lingo version 4.0 (LINDO, 1998). Hyper Lingo 4 is capable of solving a linear model that has 4000 variables and 2000 constraints. It is also capable of solving a non-linear model. However, like any other non-linear solvers, it cannot guarantee the optimality of its non-linear solver's solution. Taking into account the limitations of Hyper Lingo 4.0, a small CF problem with 4 parts, 5 machine types, a maximum of 3 machines in each cell, a maximum of 3 batches per part, and a maximum of 3 units of machines per machine type, 3 AGVs, 3 loops, 3 locations, and a maximum of 2 alternative routes between two locations, is considered. Larger CF problems are also constructed in the subsequent chapter for further experimentation with the proposed heuristics only. Using the small problem structure, ten problem instances are generated randomly for conducting the optimization experiment. Each problem instance consists of a layout of material handling system and a part production planning. To illustrate the structure of each problem instance, the same example problem presented in Chapter 5 (refer to Figure 5.1 for layout of material handling system and Table 5.2 for part production planning) is again considered here.

Based on the general structure of a small problem given above, a mathematical model described in Chapter 4 is formulated. The model resulted is a polynomial programming model or more specifically binary and general integer non-linear programming model. It has 131 integer variables and 73 constraints. The model has a non-linear objective function, 54 linear constraints and 19 non-linear constraints. Of the 131 integer variables, 99 are binary variables and 32 are general integer variables. Even though the number of variables and constraints in the model are not that many, the model could not be solved optimally just because it has non-linear terms. In order to solve for an optimal solution, linearization techniques must be used to transform the binary and general integer non-linear model into a pure binary integer linear model. Thus, first the general integer variables will have to be converted to binary integer variables and then, the resulting non-linear terms comprised of only binary integer variables will have to be

transformed into linear terms to make it a pure binary-integer linear programming model. Therefore, two types of transformation techniques are required to transform the original model.

6.1.2 Transformation Techniques

The first transformation technique transforms the general integer variables into binary variables, replacing them by their binary representations. Specifically, if the bounds on a general integer variable 'x' are

$$0 \le x \le u, \quad \text{where } 2^N \le u \le 2^{N+1},$$

then its binary representation is

$$x = \sum_{i=0}^{N} 2^i y_i,$$

where the y_i variables are (auxiliary) binary variables.

For example, suppose that the model has a general integer variable 'x' that need to be transformed into its binary representation. If x is bounded between 0 and 5, then using u = 5, its binary representation becomes $x = y_0 + 2y_1 + 4y_2$.

During the second transformation, the non-linear terms are transformed into linear terms. Two approaches could be used to perform this transformation. The first approach, which is called the standard approach, is proposed by Watters (1967). This standard approach transforms the 0-1 polynomial programming problem into a 0-1 linear programming problem by replacing each of the non-linear term or specifically the cross product term $\prod_{j \in Q} x_j$ by the new variable x_Q, and then introducing two additional constraints to the model. The constraints are:

$$(1) \quad \sum_{j \in Q} x_j - x_Q \le q - 1,$$

$$(2) \quad -\sum_{j \in Q} x_j + q x_Q \le 0,$$

$$x_Q = 0 \quad or \quad 1,$$

where q denotes the number of elements in Q.

For example, in the original model formulated above (refer to Appendix B.1), most of its non-linear terms are in the form of two or three cross product terms. Suppose that a three cross product term ($X_1.X_2.X_3$) would need to be transformed to its linear term. Also suppose that all the variables in the cross product term have been converted to binary variables. The approach starts by substituting the three-cross product term with a new binary variable X_{123}. Then, it introduces two additional constraints, (1) and (2) into the model, in the form given by

(1) $X_1 + X_2 + X_3 - X_{123} \leq 3-1$ or $X_1 + X_2 + X_3 - X_{123} \leq 2,$

(2) $- X_1 - X_2 - X_3 + 3\ X_{123} \leq 0$ or $X_1 + X_2 + X_3 - 3\ X_{123} \geq 0$

Thus, this standard approach introduces one binary variable and two linear constraints for every cross product term.

The second approach, proposed by Glover and Woolsey (1972), strives to increase the efficiency of the first approach by replacing the cross-product terms with real variables rather than binary-integer variables. Furthermore, for specific problem structures, this approach attempts to reduce the number of constraints introduced during the transformation. Specifically, it replaces the second constraint of the standard approach with its suggested constraint. Let S_j denote the set of all Q that contain the index j, and n_j denotes the number of elements of S_j. Then, for each j that appears in some set Q, (2) may alternatively be replaced by

(3) $n_j x_j \geq \sum_{Q \in S_j} x_Q$

This new constraint (3), allows x_j to be real variables rather than binary integer variables as dictated by the standard approach. However, these real variables must be bounded between 0 and 1. To illustrate the application of the second approach, suppose that the model has cross product terms of X_1X_2, X_1X_3, X_1X_4, X_2X_3, X_2X_4, $X_1X_2X_4$, and $X_2X_3X_4$. Then, S_1={12,13,14,124}, S_2={12,23,24,124,234}, S_3={13,23,234}, and S_4={14,24,124,234} with $n_1 = 4$, $n_2 = 5$, $n_3 = 3$, and $n_4 = 4$, respectively.
Thus, (3) becomes

$4X_1 \geq X_{12} + X_{13} + X_{14} + X_{124},$

$5X_2 \geq X_{12} + X_{23} + X_{24} + X_{124} + X_{234},$

$3X_3 \geq X_{13} + X_{23} + X_{234}$,

$4X_4 \geq X_{14} + X_{24} + X_{124} + X_{234}$.

where X_{12}, X_{13}, X_{14}, X_{23}, X_{24}, X_{124}, X_{234} are real variables and must be bounded between 0 and 1.

Clearly from the illustration, the second approach results in substantially fewer constraints than using the second constraint (2) of the standard approach because (2) will introduce 7 constraints, which is 3 more than the second constraint (3) proposed by the second approach. Since (3) involves only as many constraints as there are variables appearing in the cross product terms, Glover and Woolsey (1972) claim that (3) is likely to result in somewhat fewer constraints than (2), particularly if the number of cross product terms is numerous. According to them, if a standard quadratic objective function results in $n(n-1)/2$ constraints by (2), it will only result in n constraints by (3). Moreover, having real variables rather than binary variables, the model can be solved faster.

In order to find the best approach to be used in this research, both approaches would need to be compared and their results have to be analyzed to determine if they are significantly different. Hence, both approaches are applied independently to the original model formulated above. Both approaches are required to transform the original binary and general integer non-linear model into a linear model. The results obtained for both approaches are presented in Table 6.1. It compares the number of variables and constraints introduced in each of the two approaches. Notice that in the transformation above, the first and second transformation techniques must be applied sequentially.

From Table 6.1, it can be seen that in this research, the second approach proposed by Glover and Woolsey (1972) is more preferable than the standard approach. Not only the second approach reduces the number of constraints required by the model, but also has succeeded in relaxing most of the binary variables into real variables with upper bound of 1. The gain from the second approach is attributed to the number of cross product terms that appear in the original model. After applying the first transformation technique to the original model and prior to applying the second transformation, there are all together 1344 cross product terms.

Table 6.1 Comparison of the two transformation approaches

Statistic	Original Model	Standard Approach	Glover and Woolsey's approach
Total Variables	131	1507	1507
- binary integer	99	1507	163
- real, bound by 0 and 1	0	0	1344
- general integer	32	0	0
Total Constraints	70	2758	1550
- linear	51	2758	1550
- non-linear	19	0	0

This explains why the standard approach has 2758 constraints because for every cross product term, it needs two additional constraints be introduced. On the other hand, the second approach is able to take advantage of the structure of the cross product terms and thereby reduces the number of constraints that need to be introduced significantly. Furthermore, Table 6.1 reveals that the transformed model generated by the standard approach has exceeded the maximum number of constraints that can be handled by Hyper Lingo 4. Therefore, the standard approach is truly not applicable for the optimization experiment conducted here and instead, Glover and Woolsey's (1972) approach is used as the technique for the transformation of the non-linear model into linear model.

6.1.3 Illustration of the Transformation Techniques

To illustrate the process of transformation from the original model to the final model, the same example problem described earlier (refer to Figure 5.1 and Table 5.2), is used again. Based on the information given, a mathematical model is formulated as described in Chapter 4. It will be called the original model of the first problem instance and is presented in Appendix A.1. The model is a binary and general integer non-linear

programming model. It has 103 integer variables and 77 constraints. Of the 102 integer variables, 79 are binary variables and 23 are general integer variables. It has a non-linear objective function, 59 linear constraints and 18 non-linear constraints. Once the original model is obtained, all of its general integer variables have to be converted into binary variables before any non-linear terms can be transformed into linear terms. Thus, performing the first transformation technique, the original model is transformed into a pure binary integer non-linear model. The first transformed model is presented in Appendix A.2. It transforms 28 general integer variables from the original model into 56 binary variables in the first transformed model. At this time, the second transformation technique can be readily applied to the first transformed model. Doing so converts all the non-linear terms in the first transformed model into their linear counterparts. This second transformed model or the final model is now a mixed integer linear programming model. It is presented in Appendix A.3. The statistics on the number of variables and constraints involved during the first and second transformation are presented in Table 6.2

Table 6.2 Statistic of transformation for the first problem instance

Statistic	**Original Model**	**First Transformation**	**Second Transformation**
Total Variables	103	131	615
- binary integer	75	131	131
- general integer	28	0	0
- real, bound by 0 and 1	0	0	484
Total Constraints	77	77	668
- linear	59	59	668
- non-linear	18	18	0

6.1.4 Issue on the Capacity of AGV

During the optimization experiment, there is one key factor that needs to be determined by the management. That is the capacity of the AGVs. If the management sets a very restricted AGV capacity for the current planning horizon, then the mathematical model formulated would yield an infeasible solution. On the other hand, if the AGV capacity is not limited, meaning that it is unlimited, the model may not always be applicable in real industry situations. Therefore, two cases of the optimization problem are addressed: the unlimited AGV capacity case and the limited AGV capacity case. In setting the appropriate AGV capacity, the management has to consider several operational factors such as changes in customer demand, machines or tools break down, AGV cost, etc. In this research, just for the purpose of the optimization experiment, the AGV capacity for the limited case is identified by performing a preliminary experiment. The model with unlimited AGV capacity is first solved. Based on this solution, the appropriate AGV capacity for the model with limited case is determined. This process is repeated for each problem instance. Since there are ten problem instances, there are all together 20 linear models that would need to be solved by Hyper Lingo 4 (LINDO, 1998).

6.1.5 Methods of Solving the Transformed Mathematical Model

One would be tempted to think that the Hyper Lingo 4 would not have any difficulty solving the linear models above because the size of the problem is relatively very small in comparison to the problems found in industry practice. Surprisingly, of the 20 linear models that were input to Hyper Lingo, only 9 out of 20 were solved optimally within the stipulated ten-hour time limit. The results are presented in Table 6.3.

Table 6.3 Results obtained from solving the models implicitly using Hyper Lingo 4

Problem Instance	Unlimited Capacity		Limited Capacity	
	Solution	Time (Sec)	Solution	Time (Sec)
1	NS	36000	NS	36000
2	NS	36000	NS	36000
3	NS	36000	815	34202
4	NS	36000	NS	36000
5	NS	36000	NS	36000
6	699	8622	723	2544
7	723	20342	747	21210
8	NS	36000	NS	36000
9	677	5926	788	6455
10	777	26759	809	34641
Average time (sec)	27765		27905	

Note: NS stands for No Optimal Solution, which means that Hyper Lingo 4 was not able to find an optimal solution within the ten-hour time limit.

The computation time is limited to ten hours because it is sufficiently long in comparison to the time it took to solve the problems using other methods. There are two other methods that can be used to solve the problems, one is using the heuristic developed in Chapter 5 and the other is using the explicit method, which is described in the next paragraph. Using the heuristic, most of the problems are solved in less than a minute, while using the explicit method, they are solved in less than an hour. The detailed results of the heuristic and the explicit method will be presented later. Therefore, the ten-hour time limit is justifiably long to be considered as a reasonable computation time. All of the small problem instances are solved on a Pentium II 300 MHz machine with 64 MB RAM and operating under Windows NT 4.0 environment. Each model is allowed to run for ten

hours and its solution is collected at the end of the tenth hour. The result of the experiment shows that the CF problem proposed in this research is very complicated and its mathematical model is computationally difficult to solve, even for small problem instances. Even though the problem is small, it involves 131 integer variables and 484 real variables. It means that the number of explicit solutions that can be enumerated is $2^{131} = 2.722*10^{39}$, besides the 484 real variables. This explains the difficulty encountered by Hyper Lingo 4 in identifying an optimal solution. Realizing that the Hyper Lingo 4 is not able to solve the linear models within a reasonable computation time, an attempt was made to solve the model explicitly.

To solve the model explicitly, first the full model would need to be decomposed into a set of smaller models. The full model is the same identical model used by the implicit method. It is the result of transformation from the original polynomial programming model into mixed integer linear programming (MIP) model with all its real variables upper-bounded by one. Once decomposed, the set of smaller models is now used to represent the full model. Each model in the set would be solved individually using a linear solver (Hyper Lingo 4) and the best result with respect to the minimum objective function value will be identified as the optimal solution for the full model. The decomposition of the full model is justified because each small model can now be solved very quickly and easily. Thus, in contrast to the full model, which takes a long time to solve, each individual small model is solved in a matter of seconds. Furthermore, the entire set of small models is solved in much less time than the time required to solve the full model directly. The decomposition of the full model can be achieved by analyzing the structure of the full model and identifying the appropriate variables that can facilitate partitioning.

In decomposing the full model, a set of variables has to be identified and assigned specific values. By analyzing the structure of the full model, it was determined that the variable, which handles the assignment of a machine to a location, is the best choice for partitioning. This variable is selected for two reasons. First, it has just the right number of possible combinations between machines and locations. With respect to the small problem instances considered in this research, there are 1680 ways of positioning machines to their locations. The small problem structure has a maximum of 9 machines

and 3 locations. In order to find the number of different ways that 9 machines can be placed at 3 locations with a maximum of 3 machines at each location, the mathematical formula for distinct permutation is used. Thus,

$$\binom{9}{3,3,3} = \frac{9!}{3!3!3!} = 1680 \text{ ways}$$

It means that the partition will produce 1680 small models. Second, in setting this variable to a specific value, the size of the full model has been reduced significantly. For example, suppose that the general structure of a small problem is used. From Table 6.1, it can be seen that the full model has 1507 binary variables and 1550 linear constraints. In setting the above variable to a specific value, the full model has been reduced to just 280 binary variables and 355 linear constraints. This reduction is very significant, considering that a unit decrease in the number of variable translates into the power of two reduction. Therefore, it is easy to see why the revised model can now be solved efficiently by Hyper Lingo 4. On the average, each small model is solved in less than two minutes, and furthermore, the entire set of small models can be solved in less than an hour. In the mathematical model presented in Chapter 4, the variable, which handles the assignment of machine to a location, is represented by the binary variable X_{ipr}. It identifies that unit p th of machine type i is located at r. Other variables can also be chosen but they may not partition the full model efficiently and also they may not reduce the size of the full model significantly that it can be solved quickly.

Solving the linearized model explicitly is equivalent to solving 1680 small models sequentially. Each small model is characterized by the unique combination of X_{ipr} variables or machine-location configuration. So moving from one machine-location configuration to another, the algorithm will record the best solution found so far. In order to enhance the efficiency of solving the small models, the best solution found so far is also used as an upper bound. The upper bound is used to terminate the linear solver immediately when the LP-relaxation of a MIP model cannot yield an optimal solution that is smaller than the upper bound. Using this procedure would save computation time by not identifying solutions that are deemed inferior. For example, after 5 moves of the explicit search, the best solution found so far is 861. So, at the sixth move, an upper bound of 861 is introduced into the small models. If the LP-relaxation of the MIP model

cannot yield a feasible solution that is smaller than 861, the linear solver will terminate immediately and the next move will be performed. At the seventh move, the same upper bound of 861 would be used. Whenever a move identifies a better solution, the next move will use that solution as its upper bound. Performing explicit search in such a manner has resulted in saving computation time significantly.

Table 6.4 presents the results obtained from solving the 10 problem instances explicitly. Each problem instance involves two cases: the unlimited AGV capacity case and the limited AGV capacity case. All together there are 20 linear models that are solved explicitly. The average execution time for solving them is 1793 seconds or about half an hour. This execution time is far quicker than directly solving the full model, recognizing that half of the full models were not solved within the stipulated ten-hour time limit.

Table 6.4 Results obtained from solving the models explicitly

Experiment No.	**Unlimited AGV Capacity**		**Limited AGV Capacity**	
	Solution	**Time (Sec)**	**Solution**	**Time (Sec)**
1	831	1397	855	1446
2	785	1993	809	2060
3	813	1763	815	1711
4	889	2222	924	2339
5	951	2962	992	3506
6	699	1297	723	1233
7	723	1650	747	1686
8	845	1737	877	1664
9	677	1020	788	1065
10	777	1540	809	1568
Average Time (sec)	1758		1828	

Note: NS stands for No Feasible Solution

6.2 Comparison of Optimal Solution to the Solution of the Heuristic Algorithms

Once the optimal solutions are obtained, the subsequent step is to compare the solution and execution time of the optimal solution presented in Table 6.4 to the solution and execution time of the tabu search-based heuristic developed in this research. Based on the features that have significant impact on the performance of tabu search, the tabu search-based heuristic can be implemented in six different algorithms. As mentioned in Chapter 5, the features considered in this research are the tabu list size and the application of long-term memory. Two types of tabu list size can be applied: the constant and the variable tabu list size. Also, two strategies can be used for the long-term memory, one based on maximum frequency and the other based on minimum frequency. In addition, each heuristic algorithm employs two levels of search, which are executed as the inside tabu search and the outside tabu search. Thus, the six different algorithms of tabu search-based heuristic are organized as follows:

Table 6.5 The six different algorithms of the tabu search-based heuristic

Heuristic Type No.	**Inside Search**		**Outside Search**	
	Tabu List	**Memory**	**Tabu List**	**Memory**
TS1	Constant	Short	Constant	Short
TS2	Constant	Short	Constant	Long-Max
TS3	Constant	Short	Constant	Long-Min
TS4	Variable	Short	Variable	Short
TS5	Variable	Short	Variable	Long-Max
TS6	Variable	Short	Variable	Long-Min

As seen from Table 6.5, the inside tabu search does not employ the application of long-term memory. The reason, as noted before, is that the inside tabu search does not

have as much impact as the outside tabu search in determining the optimal/near-optimal solution. The search space for the inside tabu search is much smaller than that for the outside tabu search. Therefore, it may not be worthwhile to apply the long-term memory for the inside tabu search. On the other hand, the outside tabu search employs the long-term memory in TS 2,3, 5, and 6. All of the six different algorithms of the tabu search-based heuristic are tested on the same 10 problem instances generated earlier. Based on the grouping of AGV capacity, the results are presented in Table 6.6 for the unlimited case and Table 6.7 for the limited case.

Table 6.6 Test results of the six different heuristics based on unlimited AGV capacity

Problem Instance	**Tabu Search-based Heuristic Algorithms with Unlimited AGV Capacity**											
	Heuristic 1		**Heuristic 2**		**Heuristic 3**		**Heuristic 4**		**Heuristic 5**		**Heuristic 6**	
	Dev	**Time**	**Dev**	**Time**	**Dev**	**Time**	**Dev**	**Time**	**Dev**	**Time**	**Dev**	**Time**
1	3.61	21	3.61	36	0.00	35	3.61	16	3.61	35	0.00	34
2	0.00	20	0.00	38	0.00	32	1.91	26	1.91	49	1.91	50
3	8.12	9	8.12	17	2.58	14	0.00	25	0.00	43	0.00	45
4	0.00	8	0.00	16	0.00	19	0.00	20	0.00	38	0.00	53
5	0.00	26	0.00	46	0.00	39	0.00	23	0.00	41	0.00	44
6	0.00	17	0.00	27	0.00	22	0.00	16	0.00	29	0.00	29
7	0.00	11	0.00	19	0.00	19	0.00	16	0.00	31	0.00	31
8	12.66	6	12.66	10	12.66	10	0.00	29	0.00	47	0.00	44
9	12.70	11	12.70	20	0.00	32	12.70	27	12.70	50	0.00	50
10	0.00	34	0.00	43	0.00	58	7.98	20	7.98	36	0.00	40
Average	3.71	16.3	3.71	27.2	1.52	28.0	2.62	21.8	2.62	39.9	0.19	42.0

Note: Dev is the percentage deviation of the algorithm's solution from the optimal solution, while Time is the computation time of the algorithms in seconds.

Looking at Table 6.6 and Table 6.7, it is easy to see that most heuristic algorithms are capable of identifying the optimal solution. The average percentage deviation of the six heuristic algorithms is well below 4% from the optimal solution. The average

Table 6.7 Test results of the six different heuristics based on limited AGV capacity

Problem Instance	Tabu Search-based Heuristic Algorithms with Limited AGV Capacity											
	Heuristic 1		Heuristic 2		Heuristic 3		Heuristic 4		Heuristic 5		Heuristic 6	
	Dev	Time	Dev	Time	Dev	Time	Dev	Time	Dev	Time	Dev	Time
1	3.51	21	3.51	34	0.00	36	3.51	27	3.51	39	0.00	42
2	6.30	38	6.30	50	0.00	58	6.30	36	6.30	65	0.00	67
3	0.00	29	0.00	52	0.00	38	0.00	22	0.00	44	0.00	44
4	0.00	13	0.00	24	0.00	22	0.00	23	0.00	43	0.00	45
5	0.10	33	0.10	61	0.10	40	0.10	23	0.10	41	0.10	38
6	0.00	20	0.00	29	0.00	27	0.00	22	0.00	33	0.00	33
7	0.00	21	0.00	34	0.00	35	0.00	26	0.00	41	0.00	49
8	0.00	13	0.00	24	0.00	25	0.00	29	0.00	54	0.00	55
9	9.52	13	9.52	24	0.00	37	0.00	34	0.00	60	0.00	51
10	0.00	15	0.00	27	0.00	23	0.00	23	0.00	42	0.00	39
Average	1.94	21.6	1.94	35.9	0.01	34.1	0.99	26.5	0.99	46.2	0.01	46.3

Note: Dev is the percentage deviation of the algorithm's solution from the optimal solution, while Time is the execution time of the algorithms in seconds.

Table 6.8 Average Performance of the six different heuristic algorithms

Heuristic Algorithm	Average Deviation	Average Time (sec)
TS1	2.83%	18.95
TS2	2.83%	31.55
TS3	0.77%	31.05
TS4	1.81%	24.15
TS5	1.81%	43.05
TS6	0.10%	44.15

percentage deviation and computation time for both the unlimited and the limited problems with each heuristic algorithm are also presented in Table 6.8

For this small problem structure, clearly TS6 performs better than the other algorithms. With the average percentage deviation of just 0.1%, TS6 performs remarkably well. The longer average execution time for TS6 is well justified by the quality of solution it produces. The next best performer is TS3 with 0.77% deviation from the optimal solution. With the average percentage deviation of less than one percent, both algorithms are certainly very effective in obtaining optimal/near-optimal solution. An analysis of their structure reveals that both algorithms are almost similar. Both algorithms employ the use of long-term memory and their restart strategies are based on minimal frequency. Since the objective of this chapter is to assess the quality of the heuristic algorithms developed in this research, the detailed analysis of the results obtained from the six algorithms is deferred to Chapter 7.

In summary, this chapter has attempted to assess the quality of the solution produced by the heuristic algorithms developed in Chapter 5. A small problem structure was constructed in order not to exceed the limitations in Hyper Lingo 4. Then, based on the small structure, ten problem instances were generated for experimentation. Also, depending on the capacity of AGV, each problem instance was solved with its corresponding limited and unlimited AGV resources. Therefore, all together there were 20 problem instances that needed to be solved for this small problem. To obtain the optimal solution for each problem instance, a mathematical model had to be formulated based on the description presented in Chapter 4. Since the model was non-linear, a significant effort was expended to transform it into a linear model. The transformed linear model for the small problem could be solved optimally using the implicit or explicit methods. The implicit solution technique had failed terribly because half of the problems could not be solved within a reasonable computation time (refer to Table 6.3). On the other hand, the explicit solution technique had been beneficial. All the problems were solved optimally in less than an hour using the explicit method aided by the upper bound (refer to Table 6.4). A comparison with the optimal solution for each problem instance was made with the solution determined by the heuristic algorithms developed in Chapter 5. The results of the comparison are presented in Table 6.6 and Table 6.7. The average

percentage deviation for each heuristic for the combined results from both tables is presented in Table 6.8. The test results of the 10 problem instances with two cases each, showed that the heuristic algorithm has an average percentage deviation of less than 3% from the optimal solution. In addition, comparing the execution time of the heuristic to the time it takes for Hyper Lingo 4 to solve the model implicitly or explicitly, it is clear that the heuristic algorithm is extremely efficient. The heuristic solved most of the problem instances in less than two minutes, while the other methods needed one to ten hours of computation time. The result obtained concludes that the heuristic algorithms developed in this research are capable of identifying good quality solutions and therefore, should be considered in the implementation of CMS.

Recognizing the success in solving the small problem structure, the six heuristic algorithms are also used to solve medium and large problem structures as described in Chapter 7. The medium problem structure has 20 parts, a maximum of 4 operations, 6 locations, and a maximum of 24 machines in the system, while the large problem structure has 36 parts, a maximum of 5 operations, 9 locations, and a maximum of 36 machines in a system. The detailed analysis of the results for the six different heuristic algorithms for the small, medium, and large problems are presented in the next chapter as well.

7. RESULTS AND DISCUSSIONS

In Chapter 6, the optimization experimentation has shown that the heuristic algorithms developed in this research are capable of identifying good quality solutions for the small problem. In fact, one of the heuristic algorithms, which is TS6, has only missed identifying the optimal solutions in two out of the twenty problems that were solved. This implies for the test problems attempted, TS6 is capable of finding optimal solutions 90% of the time. In addition, the optimization experimentation shows that TS6 has a minimal average percentage deviation of just 0.1% from its optimal solution. This result is remarkably good considering that it only takes 44 seconds on the average to solve each problem instance. Thus, the heuristic algorithms are also used to solve medium and large problems. Even though the intent is not to determine the quality of the solutions determined for the medium and large problem instances with respect to their optimal solutions, a relevant research question is to determine which of the six heuristic algorithms would perform better when the size of the problems increases or changes. In order to answer this question, medium and large problem structures have to be constructed for experimentation as described below.

Recall from Chapter 6, the small problem structure has 5 parts, a maximum of 3 operations for each part, 5 machine types, 3 locations, 3 AGVs, and a maximum of 2 routes between any two locations. These parameters of the small problem structure will be denoted as 5P*3O*5MT*3L*3A*2R, respectively, according to the description given above. Given the size of the small problem, the medium problem structure is constructed as 20P*4O*10MT*6L*3A*2R. Its parameters are adapted from the test problem used by Boctor (1991). As for the large problem structure, it is constructed as 30P*5O*15MT*9L*5A*3R. It is taken from King (1980) and Heragu (1997). Its parameters are also modified accordingly so that it could be used in tandem AGV system. In comparison to the small problem, the complexity of the medium or large problem is estimated to be many folds higher with regard to the computation time determined from the experiment. For illustration, most of the small problems are solved in less than two minutes, but the medium problems have taken up to 12 hours of computation time to

solve, while the large problems could take as long as 44 hours to solve. The increase in computation time for medium and large problems are unavoidable because for one unit increase in the parameters of the problem structure, the complexity of the problem has raised exponentially. These parameters include the number of parts, the number of machines, the number of cells, and the number of routes. Therefore, in this research, the number of test problems used for the small, medium, and large problem structures will vary slightly. Regardless the number of test problems used, the experiment for each problem structure will strictly follow the guidelines given by the design of experiment (Montgomery, 1991). The detailed explanation of the experiment is given in section 7.3. Accordingly, the objectives of this chapter can be stated as follows:

1. To analyze the performance of the six different tabu search-based heuristics on each problem structure.
2. To analyze the impact of tabu search features, particularly the tabu list size and the long-term memory, on each problem structure.

7.1 Data Generation

Before comparing the performance of the six different heuristic algorithms, it is essential to describe the procedure used in generating the sample data for each problem structure. As mentioned in Chapter 5, the sample data for each problem structure will consist of an AGV layout and a part production plan. The part production plan is simply a combination of the part routing matrix and machine workload matrix. As the AGV layout does not change as often as the part production plan and it is also difficult to construct, only one AGV layout is specified for each problem structure. In industry practice, the AGV layout can be considered *permanent* in comparison to the number of changes performed on the part production plan as a result of changes in demand or tool breakdown. Thus, each problem structure might have several part production plans but only one AGV layout is used with all of the plans. The AGV layout for each problem structure is presented in Figure B.1, B.2, and B.3 (Appendix B) for small, medium, and large problem structures, respectively. In contrast to the predetermined AGV layout for

each problem structure, the part production plans are generated using a randomization process. The procedure used in the randomization is outlined below:

(i) All the randomization processes are set to give uniformly distributed random numbers. The random numbers will always take integer values.

(ii) Randomize the number of operations for each part between 2 and *U*. *U* is the upper bound on the number of operations, and is assigned 3, 4, and 5 for small, medium, and large problems, respectively. For example, part 1 could have 3 operations, and part 2 could have 2 operations, etc.

(iii) Randomize the assignment of a machine type to an operation of a part between 1 and *MT*. *MT* is the maximum number of machine types allowed in the system. For example, part 1 operation 1 is assigned to machine type 5, part 1 operation 2 is assigned to machine type 3, etc. If two consecutive operations of a part are assigned to the same machine, repeat the last process. For example, if operation 1 and 2 of part 1 is assigned to machine type 4, then operation 2 of part 1 has to be reassigned with a different machine type (i.e. not machine type 1).

(iv) Randomize the number of batches required of a part between 1 and *U*. *U* is the upper bound on the number of batches, and is assigned 3, 5, and 6 for small, medium, and large problems, respectively. For example, part 1 could have 3 batches, part 2 could have 1 batch, and part 3 could have 3 batches, etc.

(v) Randomize the processing time for each operation of a part between *L* and *U*. *L* is the lower bound on the random number and is assigned 5 for every problem structure, while *U* is the upper bound on the random number and is assigned 80, 50, and 40 for small, medium, and large problems, respectively. Before assigning the random number to an operation, the random number needs to be divided by 10. The unit of the processing time will be in hours. For example, part 1 operation 1 has processing time of 5/10 = 0.5 hour, part 1 operation 2 has processing time of 43/10 = 4.3 hours, etc. If the randomization process results in a processing time that is less than 0.5 hours per batch, repeat the process. The reason is that it would be impractical in real industry practice to have one batch of part with a processing time of less than 30 minutes. For example, if part 1 has 2 batches, then its operations

could not have processing time less than 1 hour, and if part 2 has 4 batches, then its operations could not have processing time less than 2 hours, etc.

(vi) If the total number of machines required by the system is not between L and U, repeat the whole process starting from step (ii). L is $((C\text{-}1) * M) + 1$, while U is $C * M$. C is the maximum number of cells allowed in the system, while M is the maximum number of machines that can be assigned to a cell. The lower bound is necessary in order to guarantee that every cell will be occupied by at least one machine. For example, in the small problem, where C is 3 and M is 3, L is 7 and U is 9. If L is less than 7, it is possible that a machine would not occupy one of the cells because the heuristic algorithm would minimize the inter-cellular movements. Thus, if the total number of machines generated by the randomization process is less than 7 or greater than 9, the whole randomization process is repeated, starting from step (ii).

(vii) If there is a machine type that is not get used, repeat the whole process starting from step (ii). For example, the small problem has 5 machine types. If there is a machine type (say 4) that does not have any workload assigned to it, the whole randomization process is repeated, starting from step (ii).

The data generated by the randomization process for the part production plan is presented in Appendix B1, B2, and B3, for small, medium, and large problem structures, respectively.

7.2 Number of Test Problems

The number of test problems used for the experiment is determined to be 20, 10, and 5 for small, medium, and large problem structures, respectively.

- For the small problem structure (5P*3O*5MT*3L*3A*2R), ten problem instances are generated. Each problem instance has two cases: the unlimited AGV capacity case and the limited AGV capacity case.
- For the medium problem structure (20P*4O*10MT*6L*3A*2R), 5 problem instances are generated. Similar to the small problem, each medium problem instance has two cases. In comparison to the small problem, the number of samples used by the

medium problem has been reduced by half due to its extensive computation time. Most small problems are solved in less than two minutes, while some medium problems have taken up to twelve hours to solve. This large variation in computation time between the two problem structures is mainly attributed to the differences in the size of problem structures and the size of the reduced integer LP problems. Recall from Chapter 5 that each evaluation of a configuration requires the heuristic algorithm to construct a reduced integer LP model to solve for the AGV routes. For the small problem, the average size of the reduced integer LP model would have only (5 parts * 4 operations * 2 routes) / 2 = 20 general integer variables, but for the medium problem, it could involve as many as (20 parts * 5 operations * 2 routes) / 2 = 100 general integer variables. Notice that the number of operations used in the formulae includes the I/O operation. Therefore, there are two factors that contribute to the increase in computational time for the medium problem. First, the difference in the size of the problem structure has increased the search space of the medium problem. This increase in search space has caused the search to consider more configurations before making a move and also, more moves are required before the search can be terminated. Second, the increase in the size of the reduced integer LP problem has increased the time it takes to perform one evaluation. For example, for a small problem, it would only take a few seconds to perform one evaluation, but for a medium problem, it could take a few minutes. Although only 10 problems are used for the medium problem structure, the corresponding β risk with α of 0.05 is approximately 0.08 and the power is $1 - \beta = 0.92$.

- Likewise, for the large problem structure (30P*5O*15MT*9L*5A*3R), only 5 problem instances are generated. However, in contrast to the small or medium problem, each large problem instance has only one case, the unlimited AGV capacity case. The reason is similar to the explanation provided for the medium problem. Some large problems have required as long as 44 hours of computation time to solve. Also, the average size of the reduced integer LP model for the large problem has involved as many as (30 parts * 6 operations * 3 routes) / 2 = 240 general integer variables. As the number of integer variables in the reduced integer LP problem is exceedingly large, in the interest of time a decision was made to not evaluate the

limited AGV capacity case for the purposes of experimentation with large problems in this research. In addition, from the experimentation performed on the medium problem, it is observed that the average execution time for the limited AGV capacity case is twice longer than that for the unlimited AGV capacity case. Therefore, if the limited AGV capacity case for the large problem were considered, it could conceivably take up to 84 hours just to solve one problem instance. With 5 problems, the large problem structure has a β risk of approximately 0.07 with $\alpha = 0.05$. Thus, it has a power of $1 - \beta = 0.93$.

7.3 Design of Experiment

To compare the performance of the heuristic algorithms, the procedure specified in the design of experiment, known as the single factor experiment, is employed. The factor in this case is characterized by each of the six heuristic algorithms and measured by the minimum total service time evaluated. The single factor experiment can be performed either as a completely randomized design or randomized block design. In a completely randomized design, it is normally assumed that the variability of results comes from a single source only. If two or more sources would affect the variability of results, they have to be blocked. Blocking these undesirable sources will increase the accuracy of the results as well as improve the sensitivity of the comparison. This blocking capability is provided by the randomized block design. Since the experiment performed here can also be affected by the structure of the test problems, the randomized block design (RBD) would be employed instead of the completely randomized design (CRD). Conducting the experiment in such a way will eliminate the influence of differences in structure of the test problems used by each problem structure. Recall that each problem structure will be experimented with several test problems. Therefore, if a difference in the performance of the heuristics is identified, it can be wholly attributed to the difference in the heuristics and not the difference between test problems. For further details on completely randomized design and randomized block designs, refer to the text by Montgomery (1991).

An analysis of variance is also performed to find if there is a significant difference among the total service time obtained for the test problems with the six heuristics. In this experiment, the significance level α, also referred to as type I error, is assumed equal to 5%. If a difference in the average total service time among the six heuristics is found, the Duncan Multiple Range Test (Montgomery, 1991) is then performed to identify which heuristics contributed to the difference. The normality assumption that is used to validate the analysis of variance is also checked using the normal probability plot of the residuals. From the plot developed for each problem structure, there is no severe indication of nonnormality, nor is there any evidence pointing to possible outliers. The normal probability plot for each problem structure is presented in Figure C.1 - C3 (Appendix C).

7.4 Experimental Results and Discussion

The experimentation described below is performed on a different computer than that was used with the small problem in Chapter 6. This research is undertaken over a period of time, when the technology has advanced significantly to a point that the computers in the laboratory were upgraded from Pentium II 300 MHz with 64 MB RAM to Pentium III 500 MHz with 128 MB RAM. As such, all of the experimental results documented in this chapter are based on the runs performed on the new machine. The operating system for the new machine is still the Windows NT 4.0 environment. The experimental results for each test problem obtained with each heuristic along with its computation time are presented in Table D.1 - D.5 (Appendix D). The analysis of variance for each problem structure is also presented in Table E.1 - E.3 (Appendix E). The summary of results for the average total service time along with the Duncan's analysis for each problem structure is shown in Table 7.1. Furthermore, to help visualize the results obtained from the Duncan's analysis, a table is created in terms of the homogeneous groups for each problem structure as shown in Table 7.2 –7.4. The "X" sign shown in the tables denotes the heuristics that do not differ significantly based on the Duncan's analysis.

Table 7.1 Summary of results obtained for the comparison of TS1 – TS6

Heuristic Algorithms	Average Total Service Time (TST)		
	Small 5P*3O*5MT* 3L*3A*2R	**Medium 20P*4O*10MT* 6L*3A*2R**	**Large 30P*5O*15MT* 9L*5A*3R**
TS1	838.75	4754.5	15999.0
TS2	838.75	4740.3	15408.2
TS3	822.90	4701.1	15999.0
TS4	832.35	4883.6	15795.4
TS5	830.20	4868.1	15743.3
TS6	817.25	4817.0	15733.6
Significant Difference?	Yes at $\alpha = 0.05$	Yes at $\alpha = 0.05$	No at $\alpha = 0.05$ Yes at $\alpha = 0.10$
TS1 & TS2	No	No	Yes
TS1 & TS3	Yes	No	No
TS1 & TS4	No	Yes	No
TS1 & TS5	No	No	No
TS1 & TS6	Yes	No	No
TS2 & TS3	Yes	No	Yes
TS2 & TS4	No	Yes	Yes
TS2 & TS5	No	Yes	No
TS2 & TS6	Yes	No	No
TS3 & TS4	No	Yes	No
TS3 & TS5	No	Yes	No
TS3 & TS6	No	No	No
TS4 & TS5	No	No	No
TS4 & TS6	Yes	No	No
TS5 & TS6	No	No	No

The results presented for the small problem structure in Table E.1 (Appendix E) show that there is a significant difference among the six heuristics at $\alpha = 0.05$. It also shows that the blocking effect of the experiment has been constructive to the analysis. If the test problem were not blocked during the experiment, it would have certainly affected the variability of the results. Since there is a significant difference among the six heuristics, Duncan's analysis is also performed (refer to Table 7.1). The summary of the results is presented in Table 7.2. It shows that there are three homogeneous groups as a result of performing the comparison.

Table 7.2 Results obtained for the small problem structure with Duncan's analysis

Heuristic	**Average Total Service Time**	**Average Execution Time (sec)**	**Homogeneous Group**		
TS6	817.25	26.35	X		
TS3	822.90	18.55	X	X	
TS5	830.20	25.65	X	X	X
TS4	832.35	14.3		X	X
TS1	838.75	11.15			X
TS2	838.75	18.75			X

TS6 is evaluated as the one having the best average total service time of 817.25. However, no significant difference among TS6, TS3, and TS5 is found. Nevertheless, TS1, TS2, and TS4 have determined a significantly larger total service time than TS6. Comparing the computation time taken by each heuristic, TS1 has the best average computation time of 11.15 seconds. The average total service time of TS1 is only 2.63% higher than TS6. At this point, TS1 appears to be very attractive. Not only TS1 spent less than half as much computation time as that required by TS6, but also has an average total service time that is only 2.63% higher than TS6. However, recognizing that the results obtained in this research address the design of CMS, the quality of the solution must

weigh more heavily than the computation time. During the design of CMS, significant effort must be expended to find a good quality solution. Any improvement that could be made during the design phase would certainly be advantageous to the performance of overall manufacturing system. Therefore, TS6 is recommended for the small problem structure. Though TS6 has the longest average execution time, it has the best performance. In comparison to the second best performer (TS3), TS6 is slightly better. TS6 has an average total service time of 0.72 % less than that of TS3.

Table 7.3 Results obtained for the medium problem structure Duncan's analysis

Heuristic	**Average Total Service Time**	**Average Execution Time (sec)**	**Homogeneous Group**		
TS3	4701.1	16134.5	X		
TS2	4740.3	14001.9	X		
TS1	4754.5	4845.6	X	X	
TS6	4817.0	13541.2	X	X	X
TS5	4868.1	13432.3		X	X
TS4	4883.6	5203.5			X

The results presented for the medium problem structure in Table E.2 (Appendix E) also show that there is a significant difference among the six heuristics at $\alpha = 0.05$. Similar to the small problem, the blocking effect in experimentation with the medium problem has proven to be useful. Duncan's analysis is performed on the six heuristics and the results are summarized in Table 7.3. Three homogeneous groups are formed as seen from the table. Identifying the best and the worst groups, and filtering out the overlapping heuristics, it is observed that TS3 and TS2 are significantly better than TS5 and TS4. TS3 has evaluated the best average total service time of 4701.1, while TS 4 has the worst average total service time of 4883.6. TS3 is approximately 4% better than TS4 in average total service time, but its average computation time is approximately 3 times longer than

that of TS4. Using the same reasoning provided for the small problem structure, TS3, the heuristic with the best average total service time, is recommended for the medium problem structure.

The results presented for the large problem structure in Table E.3 (Appendix E) show that there is no significant difference among the six heuristics at $\alpha = 0.05$. However, the probability value of the F-ratio is very close to rejecting the null hypothesis. To be exact, at $\alpha = 0.064$, the ANOVA shows that there is a significant difference among the six heuristics. Thus, rather than concluding that all six heuristics are equally good, the value of α is increased to 0.1 in order to identify the best heuristic. The results obtained from performing the Duncan's analysis are summarized in Table 7.4. It shows that there are two homogeneous groups. Filtering the overlapping heuristics between the two groups, TS2 is found to be significantly superior to TS4, TS3, and TS1. Comparing the best and the worst heuristic, TS2 is 4% better than TS1 in average total service time, but takes approximately four times longer than TS1 in average computation time. Again, using the same reasoning given in the above paragraphs, TS2 is recommended for the large problem structure.

Table 7.4 Results obtained for the large problem structure with Duncan's analysis

Heuristic	**Average Total Service Time**	**Average Execution Time (sec)**	**Homogeneous Group**	
TS2	15408.2	94074.2	X	
TS6	15733.6	82873.6	X	X
TS5	15743.2	99733.4	X	X
TS4	15795.4	30369.2		X
TS3	15999.0	101711.0		X
TS1	15999.0	19037.2		X

7.4.1 The Use of Long-Term Memory in Tabu Search-Based Heuristics

Of the six heuristics, TS2, TS3, TS5 and TS6 have employed the use of long-term memory, while TS1 and TS4 have only employed the use of short-term memory. In order to perform a fair comparison between the heuristics that employed the long-term memory and those that did not, two groups of comparison have to be made. The first group is the comparison that is restricted to the heuristics that employed fixed tabu list size. In this group, TS2 and TS3 are compared to TS1. The second group is the comparison that is restricted to the heuristics that employed variable tabu list sizes. In this group, TS5 and TS6 are compared to TS4. For the comparison, two types of measurement could be used: one based on the Duncan's test and the other based on the numerical difference test. The Duncan's test uses the result obtained from Table 7.1 to perform the comparison, while the numerical difference test compares the numerical differences between the average total service time of the two heuristics. The results obtained from the comparison according to each type of measurement are presented in Table 7.5, and can be

Table 7.5 Comparison of the heuristics that use long-term memory and those that use only short-term memory

Measurement	Group	Comparison	Problem Size		
			Small	Medium	Large
Duncan's Test	1	TS1 & TS2	-	-	No
		TS1 & TS3	No	-	-
	2	TS4 & TS5	-	-	-
		TS4 & TS6	No	-	-
Numerical Differences	1	TS1 & TS2	-	No	No
		TS1 & TS3	No	No	-
	2	TS4 & TS5	No	No	No
		TS4 & TS6	No	No	No

interpreted as follows:

- A "-" sign means that there is no significant or numerical difference between the two heuristics.
- A "Yes" means that the first heuristic performs better than the second heuristic, and a "No" means vice versa.

From Table 7.5, based on Duncan's test, 3 out of the 12 comparisons show that the heuristics with long-term memory are significantly better than the ones without it. In agreement with the results obtained from Duncan's test, the test based on the numerical difference also shows that the heuristics with long-term memory have a better average total service time in 10 out of the 12 comparisons. Thus, it can be concluded that the addition of long-term memory in tabu search improves the search significantly to identify a better solution. Although the long-term memory has not produced a better average total service time in every case attempted, it is capable of improving the search in most cases.

For the comparison of the use of long-term memory based on maximal frequency (OLTM_MAX) with the use of long-term memory based on minimal frequency (OLTM_MIN), the two types of measurement described above are used again. In this comparison, all of the heuristics that use OLTM_MAX will be compared to the heuristics that use OLTM_MIN. The results of the comparison are presented below with the same interpretation as before.

From Table 7.6, based on Duncan's test, there are three comparisons that favor the heuristics that use OLTM_MIN in contrast to one comparison that favors the heuristics that use OLTM_MAX. This shows that the OLTM_MIN is significantly better than the OLTM_MAX. As for the numerical difference test, there are 7 comparisons that favor the heuristics that use OLTM_MIN in contrast to 5 comparisons that favor the use of OLTM_MAX. Although the number of comparisons for both types of heuristics is close, the numerical difference test still indicates that the OLTM_MIN has resulted in a better average total service time than the OLTM_MAX. Therefore, based on the results of the two tests, it can be concluded that the use of long-term memory based on minimal

frequency strategy would be preferred than the use of long-term memory based on maximal frequency strategy.

Table 7.6 Comparison of the heuristics that use OLTM_MAX and OLTM_MIN

Measurement	Comparison	Problem Size		
		Small	Medium	Large
Duncan's Test	TS2 & TS3	No	-	Yes
	TS2 & TS6	No	-	-
	TS5 & TS3	-	No	-
	TS5 & TS6	-	-	-
Numerical Differences	TS2 & TS3	No	No	Yes
	TS2 & TS6	No	Yes	Yes
	TS5 & TS3	Yes	No	Yes
	TS5 & TS6	No	No	No

7.4.2. The Use of Tabu-List Size in Tabu Search-Based Heuristics.

Of the six heuristics, TS1, TS2, and TS3 employed the fixed tabu list size while TS4, TS5, and TS6 employed the variable tabu list sizes. Similar to the comparison performed for the long-term memory, three groups of comparison and two types of measurement are used to compare the performance of fixed versus variable tabu list sizes in tabu search-based heuristics. The first group consists of the comparison among the heuristics that use only the short-term memory. The second group consists of the comparison among the heuristics that use long-term memory with OLTM_MAX. Finally, the third group consists of the comparison among the heuristics that use long-term memory with OLTM_MIN. The two types of measurement are the same as before.

Table 7.7 Comparison of the heuristics that use fixed and variable tabu list sizes

Measurement	Group	Comparison	Problem Structure		
			Small	Medium	Large
Duncan's Test	1	TS1 & TS4	-	Yes	-
	2	TS2 & TS5	-	Yes	-
	3	TS3 & TS6	-	-	-
Numerical Differences	1	TS1 & TS4	No	Yes	No
	2	TS2 & TS5	No	Yes	Yes
	3	TS3 & TS6	No	Yes	No

From Table 7.7, based on the Duncan's test, the heuristics that use fixed tabu list size have two comparisons that are significantly better than those that use variable tabu list size. This shows that the Duncan's test favors the fixed tabu list over the variable tabu list. Contrary to the results obtained from the Duncan's test, the numerical difference test shows that there are 5 comparisons that favor the use of variable tabu list sizes and 4 comparisons that favor the use of fixed tabu list size. This result indicates that the use of variable tabu list sizes has resulted in a better average total service time than the use of fixed tabu list size. Even though it appears that there is a conflict of results between the two tests, the results obtained from the Duncan's test are more important than those from the numerical difference test. Two reasons are given for the above claim. First, Duncan's test is based on statistical analysis, which takes into account the variation that could possibly be introduced in the results of the heuristics. Duncan's test is certainly more accurate than the numerical difference test in performing comparison between heuristics. Second, based on the numerical difference test, the number of comparisons that favors fixed or variable tabu list sizes is very close to one another. With just one comparison difference between the two types of heuristics, either one could come up as a winner with a slight margin. Therefore, the results obtained from the Duncan's test would be used. It can be concluded that the use of fixed tabu list size is preferred over the variable tabu list sizes in tabu search-based heuristic.

In conclusion, the experimental results reveal that the use of long-term memory is essential to the tabu search algorithms developed in this research. It is recommended that the long-term memory be employed with the minimal frequency strategy. It is also observed that the application of fixed tabu list size has resulted in a better solution than the application of variable tabu list sizes. Therefore, TS3, which is the tabu search-based heuristic with fixed tabu list size and long-term memory based on minimal frequency, is recommended for solving the cell formation problem considered in this research.

8. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

In the design of CMS, one of the most important issues that need to be addressed is the cell formation (CF) problem. Although CF has received a lot of attention, very little effort has been expended to investigate the role of alternative cell locations and alternative routing of material handling equipment in its development. Research reported in the past has assumed that the location of a cell is fixed or known a priori. The capacity of the material handling equipment used in the manufacturing system has also been assumed unlimited in the CF problem. This latter assumption has led to considering a unique route between two cells instead of multiple routes. This research has presented a more accurate and comprehensive framework for the CF problem by including in its model development these two major issues as well as other important factors that are common to CF. These include machine capacity limitations, batches of part demands, non-consecutive operations of parts, and the maximum number of machines assigned to a cell.

The CF problem was first formulated as a binary and general integer non-linear programming model. The mathematical model is proven to be NP-hard in the strong sense. This rules out the possibility of employing an implicit enumeration-based algorithm such as the branch and bound technique to efficiently determine the optimal solution even on problems of moderate size. Therefore, a higher level heuristic solution algorithm, based on a concept known as tabu search, was proposed to efficiently solve the problem. The tabu search was implemented on two levels with the outside tabu search operating as the navigator for the entire search, while the inside tabu search making minor adjustments to the search process for optimal performance. Six different tabu search-based heuristics were developed by incorporating features such as the long-term memory and tabu list size. Significant effort was expended to assess the solution quality obtained from the heuristics. A small problem structure was constructed for this purpose. As the mathematical model formulated for the small problem structure is non-linear, transformation techniques were used to transform it into a linear model. The result of the transformation was a mixed integer linear programming model, where the real variables

using implicit or explicit methods. Since the implicit methods fails to solve the models within a reasonable computation time, the explicit method, aided by suitable upper bounds, has been used. A comparison is then performed between the optimal solutions and the heuristic solutions of the small problem instances. The results obtained with the heuristic solution algorithm on ten small problem instances show that the heuristics have an average percentage deviation of less than 3% from the optimal solution. TS6, the heuristic that uses variable tabu list sizes and long-term memory based on minimal frequency, has performed remarkably well with an average percentage deviation of just 0.1% from the optimal solution. This supports the premise that the heuristic algorithms developed in this research are capable of identifying good quality solutions.

The heuristic algorithms were also tested on medium and large problem structures. A single factor experiment based on randomized block design has been used to compare the performances of the six different heuristics (TS1-TS6) using the total service time as the criterion. For the small, medium, and large problem, TS6, TS3, and TS2, respectively are recommended. In general, based on the analysis of the experimental results with all problem structures, the use of long-term memory based on minimal frequency and fixed tabu list size is preferred to solve the cell formation problem in CMS.

Further research can be performed by incorporating other design factors, such as the machine utilization rate and the cost of intra-cellular movement (Heragu 1994). In this research, the utilization of machines among the machines of the same type could vary greatly in CMS. One machine could be fully used to its maximum capacity while the other machine is only used minimally. Preferably, the production manager would like to see a more balanced workload among all the machines of the same type. This is because unavoidable factors such as the operator fatigue, machine breakdown, or tool wear, could easily affect the productivity of machines when the workload of machines is not balanced. Having an appropriate utilization rate of machine would certainly reduce the overloading and bottleneck issues in CMS.

In the evaluation of material handling costs, this research has only considered the inter-cellular movement. However, as the number of machines assigned to a cell increases, the layout for the machines within a cell could become as complicated as the

layout for the cells. Therefore, this research could be extended to include the intra-cellular movement in conjunction with the inter-cellular movement.

This research could also be extended to compare the solution obtained here to the solution obtained from real time simulation. The entire investigation is performed by assuming that it is under a static environment. Although it is able to deliver a good quality solution to the CF problem, there is still a need to compare these results with that obtained in a dynamic environment. Performing a real time simulation using commercial software packages such as Pro Model or SLAM, would provide additional insights to the CF problem.

BIBLIOGRAPHY

Aljaber, N., Baek, W., and Chen, C. L., 1997, A tabu search approach to the cell formation problem. *Computers Industrial Engineering*, 32:169-185.

Askin, R. G. and Subramaniam, S. P., 1987, A cost-based heuristic for Group Technology configuration. *International Journal of Production Research*, 25:101-113.

Askin, R. G., and Vakharia, A. J., 1990, *Group Technology – cell formation and operation*, (New York: TAB Books).

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.

Boctor, F. F., 1990, A linear formulation of the machine-part cell formation problem. *International Journal of Production Research*, 29:343-356.

Bozer, Y. A., and Srinivasan, M. M., 1989, Tandem configurations for AGV systems offer simplicity and flexibility. *Industrial Engineering*, 21:23-27.

Bozer, Y. A., and Srinivasan, M. M., 1991, Tandem configurations for AGV systems and the analysis of single vehicle loops. *IIE Transactions*, 23:72-82.

Bozer, Y. A., and Srinivasan, M. M., 1992, Tandem AGV systems: a partitioning algorithm and performance comparison with conventional AGV systems. *European Journal of Production Research*, 63:173-191.

Burbidge, J. L., 1963, Production Flow Analysis. *The Production Engineer*, 42:742.

Burbidge, J. L., 1975, *The Introduction of Group Technology*, (New York: Wiley).

Burbidge, J. L., 1979, *Group Technology in the Engineering Industry* (London: Mechanical Engineering Publications).

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

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

Chen, W. H., and Srivsasta, B., 1994, Simulated annealing procedures for forming machine cell in group technology. *European Journal Operation Research*, 71:100-111.

Egbelu, P.J., and Tanchoco, J. M. A., 1984, Characterization of automated guided vehicle dispatching rules. *International Journal of Production Research*, 22:359-374.

El-Essawy, I. F. K., 1971. The development of component flow analysis as a production systems' design for multi-product engineering companies. *PhD thesis*, UMIST, U.K.

Gaither, N., 1992, *Production and Operations Management* (Orlando: The Dryden Press).

Gaskins, R. J., and Tanchoco, J. M. A., 1987, Flow path design for automated guided vehicle systems. *International Journal of Production Research*, 25:667-676.

Glover, F., 1986, Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533-549.

Glover, F., 1989, Tabu search-Part I. *ORSA Journal on Computing*, 1:190-206.

Glover, F., 1990a, Tabu search-Part II. *ORSA Journal on Computing*, 1:4-32.

Glover, F., 1990b, Tabu search: A tutorial. *Interfaces*, 20:74-94.

Glover, F., and Laguna, M. M., 1992, Tabu search. *In Modern Heuristic Techniques for Combinatorial Problems*, C. R. Reeves (ed.) (New York: Halsted Press).

Glover, F., and Woolsey, E., 1972, Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Management Science Report Series*, no. 72-3, Business Research Division, University of Colorado, Boulder, Colorado.

Goetz, W. G., and Egbelu, P. J., 1990, Guide path design and location of load pick-up/drop-off points for an automated guided vehicle system. *International Journal of Production Research*, 28:927-941.

Harvey, N., 1993, The socio-technical implementation of cellular manufacturing in American and German metal working firms. *PhD thesis*, University of Wisconsin, Madison, Wisconsin.

Heragu, S. S., 1994, Group technology and cellular manufacturing. *IEEE Transactions on Systems, Man, and Cybernetics*, 24:203-214.

Heragu, S. S., and Kakuturi, S. R., 1997, Grouping and placement of machine cells. *IIE Transactions*, 29:561-571.

Hyer, N. L., 1984, The potential of group technology for US manufacturing. *Journal of Operations Management*, 4:183-202.

Javed, M. A., 1993, Planning of manufacturing processes using artificial neural networks. *Intelligent Engineering Systems through Artificial Neural Networks*, 3:681-688.

King, J. R., 1980a, Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm. *International Journal of Production Research*, 18:213-219.

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

Kusiak, A., and Chow, W. S., 1987, Decomposition of manufacturing systems. *IEEE Transactions Robotics and Automation*, 4:457-471.

Leung, L. C., Khator, S. K., and Kimbler, D. L., 1987, Assignment of AGVS with different vehicle types. *Material Flow*, 4:65-72.

Lin, J. T., 1990, Determine how many AGVs are needed. *Industrial Engineering*, 22:53-56.

Logendran, R., Ramakrishna, 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*, 32:273-297.

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

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

Malmborg, C. J., 1990, A model for the design of zone control automated guided vehicle systems. *International Journal of Production Research*, 28:1741-1758.

Maxwell, W. L., and Muckstadt, J. A., 1982, Design of automated guided vehicle systems. *IIE Transactions*, 14:114-124.

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

Moodie, C., Uzsoy, R., and Yih, Y., 1995, *Manufacturing Cells: a systems engineering view*, (London: Taylor & Francis).

Montgomery, D.C., 1991, *Design and Analysis of Experiments* (New York: Wiley).

Puvanunt, V, 1996, The effect of cell locations in the processes of machine duplication and part subcontracting in manufacturing cell design. *MS thesis,* Oregon State University, Corvallis, Oregon.

Ransom, G. M., 1972, *Group Technology* (New York: McGraw-Hill).

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

Selim, H. M., Askin, R. G., and Vakharia, A. J., 1995, Flexibility in cellular manufacturing: a framework and measures. *Working paper 95-108,* Department of DIS, College of Business Administration, University of Florida, Gainesville, Florida.

Selim, H. M., Askin, R. G., and Vakharia, A. J., 1998, Cell formation in group technology: review, evaluation, and directions for future research. *Computers Industrial Engineering*, 34:3-20.

Hyper LINGO version 4.0, 1998 (Chicago: LINDO Systems, Inc.)

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

Vakharia, A. J. and Chang, Y. L., 1997, Cell formation in group technology: a combinatorial search approach. *International Journal of Production Research*, 35:2025-2043.

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

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

Venugopal, V., and Narendran, T. T., 1992, Cell formation in manufacturing systems through simulated annealing: an experimental evaluation. *European Journal of Operational Research*, 63:409-422.

Watters, L. J., 1967, Reduction of integer polynomial programming problems to zero-one linear programming problems. *Operations Research*, 15:1171-1174.

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

Wemmerlov, U. and Hyer, N. L., 1989, Cellular manufacturing in the US industry: a survey of users. *International Journal of Production Research*, 27:1511-1530.

Wemmerlov, U., and Johnson, D. J., 1996, Cellular manufacturing at 46 user plants: implementation experiences and performance improvements. *Working paper*, Erdman Center for Manufacturing and Technology Management, School of Business, University of Wisconsin-Madison.

APPENDICES

APPENDIX A

MATHEMATICAL FORMULATION FOR THE FIRST PROBLEM INSTANCE OF A SMALL PROBLEM

APPENDIX A.1

Original Model of the First Problem Instance

Note: This mathematical model is the result of applying the mathematical formulation described in Chapter 4 to the first problem instance of a small problem. The model is a binary and general integer non-linear programming model. It is presented in Hyper Lingo 4.0 (LINDO, 1998) format.

! OBJECTIVE FUNCTION;

MODEL:

MIN =

!Part 1 – Operation 1;
30*S1111*B111 + 46*S1112*B111 + 58*S1112*B112 + 21*S1113*B111 +
30*S1121*B111 + 46*S1122*B111 + 58*S1122*B112 + 21*S1123*B111 +
!Part 1 – Operation 2;
39*S1111*S1212*B121 + 64*S1111*S1213*B121 + 76*S1111*S1213*B122 +
33*S1112*S1211*B121 + 52*S1112*S1213*B121 + 69*S1112*S1213*B122 +
43*S1113*S1211*B121 + 70*S1113*S1211*B122 + 52*S1113*S1212*B121 + 54*S1113*S1212*B122 +
39*S1121*S1212*B121 + 64*S1121*S1213*B121 + 76*S1121*S1213*B122 +
33*S1122*S1211*B121 + 52*S1122*S1213*B121 + 69*S1122*S1213*B122 +
43*S1123*S1211*B121 + 70*S1123*S1211*B122 + 52*S1123*S1212*B121 + 54*S1123*S1212*B122 +
!Part 1 – Operation 3;
39*S1211*S1312*B131 + 64*S1211*S1313*B131 + 76*S1211*S1313*B132 +
33*S1212*S1311*B131 + 52*S1212*S1313*B131 + 69*S1212*S1313*B132 +
43*S1213*S1311*B131 + 70*S1213*S1311*B132 + 52*S1213*S1312*B131 + 54*S1213*S1312*B132 +
39*S1211*S1322*B131 + 64*S1211*S1323*B131 + 76*S1211*S1323*B132 +
33*S1212*S1321*B131 + 52*S1212*S1323*B131 + 69*S1212*S1323*B132 +
43*S1213*S1321*B131 + 70*S1213*S1321*B132 + 52*S1213*S1322*B131 + 54*S1213*S1322*B132 +
!Part 1 – Operation 4;
48*S1311*B141 + 55*S1312*B141 + 58*S1312*B142 + 18*S1313*B141 +
48*S1321*B141 + 55*S1322*B141 + 58*S1322*B142 + 18*S1323*B141 +
!Part 2 – Operation 1;
30*S2111*B211 + 46*S2112*B211 + 58*S2112*B212 + 21*S2113*B211 +
30*S2121*B211 + 46*S2122*B211 + 58*S2122*B212 + 21*S2123*B211 +
!Part 2 – Operation 2;
39*S2111*S2212*B221 + 64*S2111*S2213*B221 + 76*S2111*S2213*B222 +
33*S2112*S2211*B221 + 52*S2112*S2213*B221 + 69*S2112*S2213*B222 +
43*S2113*S2211*B221 + 70*S2113*S2211*B222 + 52*S2113*S2212*B221 + 54*S2113*S2212*B222 +
39*S2111*S2222*B221 + 64*S2111*S2223*B221 + 76*S2111*S2223*B222 +
33*S2112*S2221*B221 + 52*S2112*S2223*B221 + 69*S2112*S2223*B222 +
43*S2113*S2221*B221 + 70*S2113*S2221*B222 + 52*S2113*S2222*B221 + 54*S2113*S2222*B222 +
39*S2121*S2212*B221 + 64*S2121*S2213*B221 + 76*S2121*S2213*B222 +
33*S2122*S2211*B221 + 52*S2122*S2213*B221 + 69*S2122*S2213*B222 +
43*S2123*S2211*B221 + 70*S2123*S2211*B222 + 52*S2123*S2212*B221 + 54*S2123*S2212*B222 +
39*S2121*S2222*B221 + 64*S2121*S2223*B221 + 76*S2121*S2223*B222 +
33*S2122*S2221*B221 + 52*S2122*S2223*B221 + 69*S2122*S2223*B222 +
43*S2123*S2221*B221 + 70*S2123*S2221*B222 + 52*S2123*S2222*B221 + 54*S2123*S2222*B222 +
!Part 2 – Operation 3;
48*S2211*B231 + 55*S2212*B231 + 58*S2212*B232 + 18*S2213*B231 +
48*S2221*B231 + 55*S2222*B231 + 58*S2222*B232 + 18*S2223*B231 +
!Part 3 – Operation 1;
30*S3111*B311 + 46*S3112*B311 + 58*S3112*B312 + 21*S3113*B311 +
30*S3121*B311 + 46*S3122*B311 + 58*S3122*B312 + 21*S3123*B311 +
!Part 3 – Operation 2;
39*S3111*S3212*B321 + 64*S3111*S3213*B321 + 76*S3111*S3213*B322 +
33*S3112*S3211*B321 + 52*S3112*S3213*B321 + 69*S3112*S3213*B322 +
43*S3113*S3211*B321 + 70*S3113*S3211*B322 + 52*S3113*S3212*B321 + 54*S3113*S3212*B322 +
39*S3121*S3212*B321 + 64*S3121*S3213*B321 + 76*S3121*S3213*B322 +
33*S3122*S3211*B321 + 52*S3122*S3213*B321 + 69*S3122*S3213*B322 +

```
43*S3123*S3211*B321 + 70*S3123*S3211*B322 + 52*S3123*S3212*B321 + 54*S3123*S3212*B322 +
!Part 3 – Operation 3;
39*S3211*S3312*B331 + 64*S3211*S3313*B331 + 76*S3211*S3313*B332 +
33*S3212*S3311*B331 + 52*S3212*S3313*B331 + 69*S3212*S3313*B332 +
43*S3213*S3311*B331 + 70*S3213*S3311*B332 + 52*S3213*S3312*B331 + 54*S3213*S3312*B332 +
39*S3211*S3322*B331 + 64*S3211*S3323*B331 + 76*S3211*S3323*B332 +
33*S3212*S3321*B331 + 52*S3212*S3323*B331 + 69*S3212*S3323*B332 +
43*S3213*S3321*B331 + 70*S3213*S3321*B332 + 52*S3213*S3322*B331 + 54*S3213*S3322*B332 +
!Part 3 – Operation 4;
48*S3311*B341 + 55*S3312*B341 + 58*S3312*B342 + 18*S3313*B341 +
48*S3321*B341 + 55*S3322*B341 + 58*S3322*B342 + 18*S3323*B341 +
!Part 4 – Operation 1;
30*S4111*B411 + 46*S4112*B411 + 58*S4112*B412 + 21*S4113*B411 +
30*S4121*B411 + 46*S4122*B411 + 58*S4122*B412 + 21*S4123*B411 +
!Part 4 – Operation 2;
39*S4111*S4212*B421 + 64*S4111*S4213*B421 + 76*S4111*S4213*B422 +
33*S4112*S4211*B421 + 52*S4112*S4213*B421 + 69*S4112*S4213*B422 +
43*S4113*S4211*B421 + 70*S4113*S4211*B422 + 52*S4113*S4212*B421 + 54*S4113*S4212*B422 +
39*S4121*S4212*B421 + 64*S4121*S4213*B421 + 76*S4121*S4213*B422 +
33*S4122*S4211*B421 + 52*S4122*S4213*B421 + 69*S4122*S4213*B422 +
43*S4123*S4211*B421 + 70*S4123*S4211*B422 + 52*S4123*S4212*B421 + 54*S4123*S4212*B422 +
!Part 4 – Operation 3;
48*S4211*B431 + 55*S4212*B431 + 58*S4212*B432 + 18*S4213*B431;
```

!End of OBJECTIVE FUNCTION;

!OPERATIONAL CONSTRAINTS;

```
!First Constraint - No Split Operations;
S1111+ S1112+ S1113+ S1121+ S1122+ S1123 = 1;
S1211+ S1212+ S1213 = 1;
S1311+ S1312+ S1313+ S1321+ S1322+ S1323 = 1;
S2111+ S2112+ S2113+ S2121+ S2122+ S2123 = 1;
S2211+ S2212+ S2213+ S2221+ S2222+ S2223 = 1;
S3111+ S3112+ S3113+ S3121+ S3122+ S3123 = 1;
S3211+ S3212+ S3213 = 1;
S3311+ S3312+ S3313+ S3321+ S3322+ S3323 = 1;
S4111+ S4112+ S4113+ S4121+ S4122+ S4123 = 1;
S4211+ S4212+ S4213 = 1;

!Second Constraint - Only One Unique Machine In Each Location;
X111+ X112+ X113 = 1;
X121+ X122+ X123 = 1;
X211+ X212+ X213 = 1;
X221+ X222+ X223 = 1;
X311+ X312+ X313 = 1;
X411+ X412+ X413 = 1;
X421+ X422+ X423 = 1;
X511+ X512+ X513 = 1;

!Third Constraint - Machine Capacity Limitations;
8*X111 - 4.0*S1111 - 7.0*S4111 >= 0;
8*X112 - 4.0*S1112 - 7.0*S4112 >= 0;
8*X113 - 4.0*S1113 - 7.0*S4113 >= 0;
8*X121 - 4.0*S1121 - 7.0*S4121 >= 0;
8*X122 - 4.0*S1122 - 7.0*S4122 >= 0;
8*X123 - 4.0*S1123 - 7.0*S4123 >= 0;
8*X211 - 3.5*S2111 - 4.0*S3111 - 6.0*S3311 >= 0;
8*X212 - 3.5*S2112 - 4.0*S3112 - 6.0*S3312 >= 0;
8*X213 - 3.5*S2113 - 4.0*S3113 - 6.0*S3313 >= 0;
8*X221 - 3.5*S2121 - 4.0*S3121 - 6.0*S3321 >= 0;
8*X222 - 3.5*S2122 - 4.0*S3122 - 6.0*S3322 >= 0;
8*X223 - 3.5*S2123 - 4.0*S3123 - 6.0*S3323 >= 0;
8*X311 - 5.0*S1211 - 3.0*S4211 >= 0;
8*X312 - 5.0*S1212 - 3.0*S4212 >= 0;
8*X313 - 5.0*S1213 - 3.0*S4213 >= 0;
8*X411 - 6.0*S1311 - 4.5*S2211 >= 0;
8*X412 - 6.0*S1312 - 4.5*S2212 >= 0;
8*X413 - 6.0*S1313 - 4.5*S2213 >= 0;
```

```
8*X421 - 6.0*S1321 - 4.5*S2221 >= 0;
8*X422 - 6.0*S1322 - 4.5*S2222 >= 0;
8*X423 - 6.0*S1323 - 4.5*S2223 >= 0;
8*X511 - 5.0*S3211 >= 0;
8*X512 - 5.0*S3212 >= 0;
8*X513 - 5.0*S3213 >= 0;

!Fourth Constraint - Maximum Number of Machines in a Location;
X111+ X121+ X211+ X221+ X311+ X411+ X421+ X511 <= 3;
X112+ X122+ X212+ X222+ X312+ X412+ X422+ X512 <= 3;
X113+ X123+ X213+ X223+ X313+ X413+ X423+ X513 <= 3;

!Fifth Constraint - Demand for each part;
!Part 1 - Operation 1;
S1111*B111 + S1112*B111 + S1112*B112 + S1113*B111 +
S1121*B111 + S1122*B111 + S1122*B112 + S1123*B111  = 3;
!Part 1 - Operation 2;
S1111*S1211*B121 + S1111*S1212*B121 + S1111*S1213*B121 + S1111*S1213*B122 +
S1112*S1211*B121 + S1112*S1212*B121 + S1112*S1213*B121 + S1112*S1213*B122 +
S1113*S1211*B121 + S1113*S1211*B122 + S1113*S1212*B121 + S1113*S1212*B122 + S1113*S1213*B121 +
S1121*S1211*B121 + S1121*S1212*B121 + S1121*S1213*B121 + S1121*S1213*B122 + .
S1122*S1211*B121 + S1122*S1212*B121 + S1122*S1213*B121 + S1122*S1213*B122 +
S1123*S1211*B121 + S1123*S1211*B122 + S1123*S1212*B121 + S1123*S1212*B122 + S1123*S1213*B121  = 3;
!Part 1 - Operation 3;
S1211*S1311*B131 + S1211*S1312*B131 + S1211*S1313*B131 + S1211*S1313*B132 +
S1212*S1311*B131 + S1212*S1312*B131 + S1212*S1313*B131 + S1212*S1313*B132 +
S1213*S1311*B131 + S1213*S1311*B132 + S1213*S1312*B131 + S1213*S1312*B132 + S1213*S1313*B131 +
S1211*S1321*B131 + S1211*S1322*B131 + S1211*S1323*B131 + S1211*S1323*B132 +
S1212*S1321*B131 + S1212*S1322*B131 + S1212*S1323*B131 + S1212*S1323*B132 +
S1213*S1321*B131 + S1213*S1321*B132 + S1213*S1322*B131 + S1213*S1322*B132 + S1213*S1323*B131  = 3;
!Part 1 - Operation 4;
S1311*B141 + S1312*B141 + S1312*B142 + S1313*B141 +
S1321*B141 + S1322*B141 + S1322*B142 + S1323*B141  = 3;
!Part 2 - Operation 1;
S2111*B211 + S2112*B211 + S2112*B212 + S2113*B211 +
S2121*B211 + S2122*B211 + S2122*B212 + S2123*B211  = 2;
!Part 2 - Operation 2;
S2111*S2211*B221 + S2111*S2212*B221 + S2111*S2213*B221 + S2111*S2213*B222 +
S2112*S2211*B221 + S2112*S2212*B221 + S2112*S2213*B221 + S2112*S2213*B222 +
S2113*S2211*B221 + S2113*S2211*B222 + S2113*S2212*B221 + S2113*S2212*B222 + S2113*S2213*B221 +
S2111*S2221*B221 + S2111*S2222*B221 + S2111*S2223*B221 + S2111*S2223*B222 +
S2112*S2221*B221 + S2112*S2222*B221 + S2112*S2223*B221 + S2112*S2223*B222 +
S2113*S2221*B221 + S2113*S2221*B222 + S2113*S2222*B221 + S2113*S2222*B222 + S2113*S2223*B221 +
S2121*S2211*B221 + S2121*S2212*B221 + S2121*S2213*B221 + S2121*S2213*B222 +
S2122*S2211*B221 + S2122*S2212*B221 + S2122*S2213*B221 + S2122*S2213*B222 +
S2123*S2211*B221 + S2123*S2211*B222 + S2123*S2212*B221 + S2123*S2212*B222 + S2123*S2213*B221 +
S2121*S2221*B221 + S2121*S2222*B221 + S2121*S2223*B221 + S2121*S2223*B222 +
S2122*S2221*B221 + S2122*S2222*B221 + S2122*S2223*B221 + S2122*S2223*B222 +
S2123*S2221*B221 + S2123*S2221*B222 + S2123*S2222*B221 + S2123*S2222*B222 + S2123*S2223*B221  = 2;
!Part 2 - Operation 3;
S2211*B231 + S2212*B231 + S2212*B232 + S2213*B231 +
S2221*B231 + S2222*B231 + S2222*B232 + S2223*B231  = 2;
!Part 3 - Operation 1;
S3111*B311 + S3112*B311 + S3112*B312 + S3113*B311 +
S3121*B311 + S3122*B311 + S3122*B312 + S3123*B311  = 3;
!Part 3 - Operation 2;
S3111*S3211*B321 + S3111*S3212*B321 + S3111*S3213*B321 + S3111*S3213*B322 +
S3112*S3211*B321 + S3112*S3212*B321 + S3112*S3213*B321 + S3112*S3213*B322 +
S3113*S3211*B321 + S3113*S3211*B322 + S3113*S3212*B321 + S3113*S3212*B322 + S3113*S3213*B321 +
S3121*S3211*B321 + S3121*S3212*B321 + S3121*S3213*B321 + S3121*S3213*B322 +
S3122*S3211*B321 + S3122*S3212*B321 + S3122*S3213*B321 + S3122*S3213*B322 +
S3123*S3211*B321 + S3123*S3211*B322 + S3123*S3212*B321 + S3123*S3212*B322 + S3123*S3213*B321  = 3;
!Part 3 - Operation 3;
S3211*S3311*B331 + S3211*S3312*B331 + S3211*S3313*B331 + S3211*S3313*B332 +
S3212*S3311*B331 + S3212*S3312*B331 + S3212*S3313*B331 + S3212*S3313*B332 +
S3213*S3311*B331 + S3213*S3311*B332 + S3213*S3312*B331 + S3213*S3312*B332 + S3213*S3313*B331 +
S3211*S3321*B331 + S3211*S3322*B331 + S3211*S3323*B331 + S3211*S3323*B332 +
S3212*S3321*B331 + S3212*S3322*B331 + S3212*S3323*B331 + S3212*S3323*B332 +
S3213*S3321*B331 + S3213*S3321*B332 + S3213*S3322*B331 + S3213*S3322*B332 + S3213*S3323*B331  = 3;
!Part 3 - Operation 4;
```

```
S3311*B341 + S3312*B341 + S3312*B342 + S3313*B341 +
S3321*B341 + S3322*B341 + S3322*B342 + S3323*B341  = 3;
!Part 4 - Operation 1;
S4111*B411 + S4112*B411 + S4112*B412 + S4113*B411 +
S4121*B411 + S4122*B411 + S4122*B412 + S4123*B411  = 2;
!Part 4 - Operation 2;
S4111*S4211*B421 + S4111*S4212*B421 + S4111*S4213*B421 + S4111*S4213*B422 +
S4112*S4211*B421 + S4112*S4212*B421 + S4112*S4213*B421 + S4112*S4213*B422 +
S4113*S4211*B421 + S4113*S4211*B422 + S4113*S4212*B421 + S4113*S4212*B422 + S4113*S4213*B421 +
S4121*S4211*B421 + S4121*S4212*B421 + S4121*S4213*B421 + S4121*S4213*B422 +
S4122*S4211*B421 + S4122*S4212*B421 + S4122*S4213*B421 + S4122*S4213*B422 +
S4123*S4211*B421 + S4123*S4211*B422 + S4123*S4212*B421 + S4123*S4212*B422 + S4123*S4213*B421  = 2;
!Part 4 - Operation 3;
S4211*B431 + S4212*B431 + S4212*B432 + S4213*B431  = 2;

!Sixth Constraint - Demand for each AGV;

!AGV 1;
48*S1111*B111 + 48*S1112*B111 + 8*S1112*B112 + 8*S1113*B111 + 48*S1121*B111 +
48*S1122*B111 + 8*S1122*B112 + 8*S1123*B111 + 48*S2111*B211 + 48*S2112*B211 +
8*S2112*B212 + 8*S2113*B211 + 48*S2121*B211 + 48*S2122*B211 + 8*S2122*B212 +
8*S2123*B211 + 48*S3111*B311 + 48*S3112*B311 + 8*S3112*B312 + 8*S3113*B311 +
48*S3121*B311 + 48*S3122*B311 + 8*S3122*B312 + 8*S3123*B311 + 48*S4111*B411 +
48*S4112*B411 + 8*S4112*B412 + 8*S4113*B411 + 48*S4121*B411 + 48*S4122*B411 +
8*S4122*B412 + 8*S4123*B411 + 48*S1311*B141 + 8*S1312*B141 + 48*S1312*B142 +
8*S1313*B141 + 48*S1321*B141 + 8*S1322*B141 + 48*S1322*B142 + 8*S1323*B141 +
48*S2211*B231 + 8*S2212*B231 + 48*S2212*B232 + 8*S2213*B231 + 48*S2221*B231 +
8*S2222*B231 + 48*S2222*B232 + 8*S2223*B231 + 48*S3311*B341 + 8*S3312*B341 +
48*S3312*B342 + 8*S3313*B341 + 48*S3321*B341 + 8*S3322*B341 + 48*S3322*B342 +
8*S3323*B341 + 48*S4211*B431 + 8*S4212*B431 + 48*S4212*B432 + 8*S4213*B431 +
41*S1111*S1212*B121 + 51*S1111*S1213*B121 + 41*S1111*S1213*B122 + 41*S1112*S1211*B121 +
46*S1112*S1213*B122 + 51*S1113*S1211*B121 + 41*S1113*S1211*B122 + 46*S1113*S1212*B122 +
41*S1121*S1212*B121 + 51*S1121*S1213*B121 + 41*S1121*S1213*B122 + 41*S1122*S1211*B121 +
46*S1122*S1213*B122 + 51*S1123*S1211*B121 + 41*S1123*S1211*B122 + 46*S1123*S1212*B122 +
41*S1211*S1312*B131 + 51*S1211*S1313*B131 + 41*S1211*S1313*B132 + 41*S1212*S1311*B131 +
46*S1212*S1313*B132 + 51*S1213*S1311*B131 + 41*S1213*S1311*B132 + 46*S1213*S1312*B132 +
41*S1211*S1322*B131 + 51*S1211*S1323*B131 + 41*S1211*S1323*B132 + 41*S1212*S1321*B131 +
46*S1212*S1323*B132 + 51*S1213*S1321*B131 + 41*S1213*S1321*B132 + 46*S1213*S1322*B132 +
41*S2111*S2212*B221 + 51*S2111*S2213*B221 + 41*S2111*S2213*B222 + 41*S2112*S2211*B221 +
46*S2112*S2213*B222 + 51*S2113*S2211*B221 + 41*S2113*S2211*B222 + 46*S2113*S2212*B222 +
41*S2111*S2222*B221 + 51*S2111*S2223*B221 + 41*S2111*S2223*B222 + 41*S2112*S2221*B221 +
46*S2112*S2223*B222 + 51*S2113*S2221*B221 + 41*S2113*S2221*B222 + 46*S2113*S2222*B222 +
41*S2121*S2212*B221 + 51*S2121*S2213*B221 + 41*S2121*S2213*B222 + 41*S2122*S2211*B221 +
46*S2122*S2213*B222 + 51*S2123*S2211*B221 + 41*S2123*S2211*B222 + 46*S2123*S2212*B222 +
41*S2121*S2222*B221 + 51*S2121*S2223*B221 + 41*S2121*S2223*B222 + 41*S2122*S2221*B221 +
46*S2122*S2223*B222 + 51*S2123*S2221*B221 + 41*S2123*S2221*B222 + 46*S2123*S2222*B222 +
41*S3111*S3212*B321 + 51*S3111*S3213*B321 + 41*S3111*S3213*B322 + 41*S3112*S3211*B321 +
46*S3112*S3213*B322 + 51*S3113*S3211*B321 + 41*S3113*S3211*B322 + 46*S3113*S3212*B322 +
41*S3121*S3212*B321 + 51*S3121*S3213*B321 + 41*S3121*S3213*B322 + 41*S3122*S3211*B321 +
46*S3122*S3213*B322 + 51*S3123*S3211*B321 + 41*S3123*S3211*B322 + 46*S3123*S3212*B322 +
41*S3211*S3312*B331 + 51*S3211*S3313*B331 + 41*S3211*S3313*B332 + 41*S3212*S3311*B331 +
46*S3212*S3313*B332 + 51*S3213*S3311*B331 + 41*S3213*S3311*B332 + 46*S3213*S3312*B332 +
41*S3211*S3322*B331 + 51*S3211*S3323*B331 + 41*S3211*S3323*B332 + 41*S3212*S3321*B331 +
46*S3212*S3323*B332 + 51*S3213*S3321*B331 + 41*S3213*S3321*B332 + 46*S3213*S3322*B332 +
41*S4111*S4212*B421 + 51*S4111*S4213*B421 + 41*S4111*S4213*B422 + 41*S4112*S4211*B421 +
46*S4112*S4213*B422 + 51*S4113*S4211*B421 + 41*S4113*S4211*B422 + 46*S4113*S4212*B422 +
41*S4121*S4212*B421 + 51*S4121*S4213*B421 + 41*S4121*S4213*B422 + 41*S4122*S4211*B421 +
46*S4122*S4213*B422 + 51*S4123*S4211*B421 + 41*S4123*S4211*B422 + 46*S4123*S4212*B422  <= 10000;

!AGV 2;
16*S1112*B111 + 32*S1112*B112 + 16*S1122*B111 + 32*S1122*B112 + 16*S2112*B211 +
32*S2112*B212 + 16*S2122*B211 + 32*S2122*B212 + 16*S3112*B311 + 32*S3112*B312 +
16*S3122*B311 + 32*S3122*B312 + 16*S4112*B411 + 32*S4112*B412 + 16*S4122*B411 +
32*S4122*B412 + 32*S1312*B141 + 16*S1312*B142 + 32*S1322*B141 + 16*S1322*B142 +
32*S2212*B231 + 16*S2212*B232 + 32*S2222*B231 + 16*S2222*B232 + 32*S3312*B341 +
16*S3312*B342 + 32*S3322*B341 + 16*S3322*B342 + 32*S4212*B431 + 16*S4212*B432 +
16*S1111*S1212*B121 + 33*S1111*S1213*B122 +
16*S1112*S1211*B121 + 32*S1112*S1213*B121 + 16*S1112*S1213*B122 + 33*S1113*S1211*B122 +
32*S1113*S1212*B121 + 16*S1113*S1212*B122 + 16*S1121*S1212*B121 + 33*S1121*S1213*B122 +
16*S1122*S1211*B121 + 32*S1122*S1213*B121 + 16*S1122*S1213*B122 + 33*S1123*S1211*B122 +
```

32*S1123*S1212*B121 + 16*S1123*S1212*B122 + 16*S1211*S1312*B131 + 33*S1211*S1313*B132 +
16*S1212*S1311*B131 + 32*S1212*S1313*B131 + 16*S1212*S1313*B132 + 33*S1213*S1311*B132 +
32*S1213*S1312*B131 + 16*S1213*S1312*B132 + 16*S1211*S1322*B131 + 33*S1211*S1323*B132 +
16*S1212*S1321*B131 + 32*S1212*S1323*B131 + 16*S1212*S1323*B132 + 33*S1213*S1321*B132 +
32*S1213*S1322*B131 + 16*S1213*S1322*B132 + 16*S2111*S2212*B221 + 33*S2111*S2213*B222 +
16*S2112*S2211*B221 + 32*S2112*S2213*B221 + 16*S2112*S2213*B222 + 33*S2113*S2211*B222 +
32*S2113*S2212*B221 + 16*S2113*S2212*B222 + 16*S2111*S2222*B221 + 33*S2111*S2223*B222 +
16*S2112*S2221*B221 + 32*S2112*S2223*B221 + 16*S2112*S2223*B222 + 33*S2113*S2221*B222 +
32*S2113*S2222*B221 + 16*S2113*S2222*B222 + 16*S2121*S2212*B221 + 33*S2121*S2213*B222 +
16*S2122*S2211*B221 + 32*S2122*S2213*B221 + 16*S2122*S2213*B222 + 33*S2123*S2211*B222 +
32*S2123*S2212*B221 + 16*S2123*S2212*B222 + 16*S2121*S2222*B221 + 33*S2121*S2223*B222 +
16*S2122*S2221*B221 + 32*S2122*S2223*B221 + 16*S2122*S2223*B222 + 33*S2123*S2221*B222 +
32*S2123*S2222*B221 + 16*S2123*S2222*B222 + 16*S3111*S3212*B321 + 33*S3111*S3213*B322 +
16*S3112*S3211*B321 + 32*S3112*S3213*B321 + 16*S3112*S3213*B322 + 33*S3113*S3211*B322 +
32*S3113*S3212*B321 + 16*S3113*S3212*B322 + 16*S3121*S3212*B321 + 33*S3121*S3213*B322 +
16*S3122*S3211*B321 + 32*S3122*S3213*B321 + 16*S3122*S3213*B322 + 33*S3123*S3211*B322 +
32*S3123*S3212*B321 + 16*S3123*S3212*B322 + 16*S3211*S3312*B331 + 33*S3211*S3313*B332 +
16*S3212*S3311*B331 + 32*S3212*S3313*B331 + 16*S3212*S3313*B332 + 33*S3213*S3311*B332 +
32*S3213*S3312*B331 + 16*S3213*S3312*B332 + 16*S3211*S3322*B331 + 33*S3211*S3323*B332 +
16*S3212*S3321*B331 + 32*S3212*S3323*B331 + 16*S3212*S3323*B332 + 33*S3213*S3321*B332 +
32*S3213*S3322*B331 + 16*S3213*S3322*B332 + 16*S4111*S4212*B421 + 33*S4111*S4213*B422 +
16*S4112*S4211*B421 + 32*S4112*S4213*B421 + 16*S4112*S4213*B422 + 33*S4121*S4213*B422 +
32*S4113*S4212*B421 + 16*S4113*S4212*B422 + 16*S4121*S4212*B421 + 33*S4121*S4213*B422 +
16*S4122*S4211*B421 + 32*S4122*S4213*B421 + 16*S4122*S4213*B422 + 33*S4123*S4211*B422 +
32*S4123*S4212*B421 + 16*S4123*S4212*B422 <= 10000;

!AGV 3;
33*S1112*B112 + 16*S1113*B111 + 33*S1122*B112 + 16*S1123*B111 + 33*S2112*B212 +
16*S2113*B211 + 33*S2122*B212 + 16*S2123*B211 + 33*S3112*B312 + 16*S3113*B311 +
33*S3122*B312 + 16*S3123*B311 + 33*S4112*B412 + 16*S4113*B411 + 33*S4122*B412 +
16*S4123*B411 + 33*S1312*B141 + 16*S1313*B141 + 33*S1322*B141 + 16*S1323*B141 +
33*S2212*B231 + 16*S2213*B231 + 33*S2222*B231 + 16*S2223*B231 + 33*S3312*B341 +
16*S3313*B341 + 33*S3322*B341 + 16*S3323*B341 + 33*S4212*B431 + 16*S4213*B431 +
16*S1111*S1213*B121 + 32*S1111*S1213*B122 +
32*S1112*S1213*B121 + 16*S1112*S1213*B122 + 16*S1113*S1211*B121 + 32*S1113*S1211*B122 +
32*S1113*S1212*B121 + 16*S1113*S1212*B122 + 16*S1121*S1213*B121 + 32*S1121*S1213*B122 +
32*S1122*S1213*B121 + 16*S1122*S1213*B122 + 16*S1123*S1211*B121 + 32*S1123*S1211*B122 +
32*S1123*S1212*B121 + 16*S1123*S1212*B122 + 16*S1211*S1313*B131 + 32*S1211*S1313*B132 +
32*S1212*S1313*B131 + 16*S1212*S1313*B132 + 16*S1213*S1311*B131 + 32*S1213*S1311*B132 +
32*S1213*S1312*B131 + 16*S1213*S1312*B132 + 16*S1211*S1323*B131 + 32*S1211*S1323*B132 +
32*S1212*S1323*B131 + 16*S1212*S1323*B132 + 16*S1213*S1321*B131 + 32*S1213*S1321*B132 +
32*S1213*S1322*B131 + 16*S1213*S1322*B132 + 16*S2111*S2213*B221 + 32*S2111*S2213*B222 +
32*S2112*S2213*B221 + 16*S2112*S2213*B222 + 16*S2113*S2211*B221 + 32*S2113*S2211*B222 +
32*S2113*S2212*B221 + 16*S2113*S2212*B222 + 16*S2111*S2223*B221 + 32*S2111*S2223*B222 +
32*S2112*S2223*B221 + 16*S2112*S2223*B222 + 16*S2113*S2221*B221 + 32*S2113*S2221*B222 +
32*S2113*S2222*B221 + 16*S2113*S2222*B222 + 16*S2121*S2213*B221 + 32*S2121*S2213*B222 +
32*S2122*S2213*B221 + 16*S2122*S2213*B222 + 16*S2123*S2211*B221 + 32*S2123*S2211*B222 +
32*S2123*S2212*B221 + 16*S2123*S2212*B222 + 16*S2121*S2223*B221 + 32*S2121*S2223*B222 +
32*S2122*S2223*B221 + 16*S2122*S2223*B222 + 16*S2123*S2221*B221 + 32*S2123*S2221*B222 +
32*S2123*S2222*B221 + 16*S2123*S2222*B222 + 16*S3111*S3213*B321 + 32*S3111*S3213*B322 +
32*S3112*S3213*B321 + 16*S3112*S3213*B322 + 16*S3113*S3211*B321 + 32*S3113*S3211*B322 +
32*S3113*S3212*B321 + 16*S3113*S3212*B322 + 16*S3121*S3213*B321 + 32*S3121*S3213*B322 +
32*S3122*S3213*B321 + 16*S3122*S3213*B322 + 16*S3123*S3211*B321 + 32*S3123*S3211*B322 +
32*S3123*S3212*B321 + 16*S3123*S3212*B322 + 16*S3211*S3313*B331 + 32*S3211*S3313*B332 +
32*S3212*S3313*B331 + 16*S3212*S3313*B332 + 16*S3213*S3311*B331 + 32*S3213*S3311*B332 +
32*S3213*S3312*B331 + 16*S3213*S3312*B332 + 16*S3211*S3323*B331 + 32*S3211*S3323*B332 +
32*S3212*S3323*B331 + 16*S3212*S3323*B332 + 16*S3213*S3321*B331 + 32*S3213*S3321*B332 +
32*S3213*S3322*B331 + 16*S3213*S3322*B332 + 16*S4111*S4213*B421 + 32*S4111*S4213*B422 +
32*S4112*S4213*B421 + 16*S4112*S4213*B422 + 16*S4113*S4211*B421 + 32*S4113*S4211*B422 +
32*S4113*S4212*B421 + 16*S4113*S4212*B422 + 16*S4121*S4213*B421 + 32*S4121*S4213*B422 +
32*S4122*S4213*B421 + 16*S4122*S4213*B422 + 16*S4123*S4211*B421 + 32*S4123*S4211*B422 +
32*S4123*S4212*B421 + 16*S4123*S4212*B422 <= 10000;

!End of OPERATIONAL CONSTRAINTS;

```
!Declare the S Variables as Binary Integer Variables;
@BIN(S1111);  @BIN(S1112);  @BIN(S1113);  @BIN(S1121);  @BIN(S1122);  @BIN(S1123);  @BIN(S1211);
@BIN(S1212);  @BIN(S1213);  @BIN(S1311);  @BIN(S1312);  @BIN(S1313);  @BIN(S1321);  @BIN(S1322);
@BIN(S1323);  @BIN(S2111);  @BIN(S2112);  @BIN(S2113);  @BIN(S2121);  @BIN(S2122);  @BIN(S2123);
@BIN(S2211);  @BIN(S2212);  @BIN(S2213);  @BIN(S2221);  @BIN(S2222);  @BIN(S2223);  @BIN(S3111);
@BIN(S3112);  @BIN(S3113);  @BIN(S3121);  @BIN(S3122);  @BIN(S3123);  @BIN(S3211);  @BIN(S3212);
@BIN(S3213);  @BIN(S3311);  @BIN(S3312);  @BIN(S3313);  @BIN(S3321);  @BIN(S3322);  @BIN(S3323);
@BIN(S4111);  @BIN(S4112);  @BIN(S4113);  @BIN(S4121);  @BIN(S4122);  @BIN(S4123);  @BIN(S4211);
@BIN(S4212);  @BIN(S4213);

!Declare the B Variables as General Integer Variables;
@GIN(B111);  @GIN(B112);  @GIN(B121);  @GIN(B122);  @GIN(B131);  @GIN(B132);  @GIN(B141);
@GIN(B142);  @GIN(B211);  @GIN(B212);  @GIN(B221);  @GIN(B222);  @GIN(B231);  @GIN(B232);
@GIN(B311);  @GIN(B312);  @GIN(B321);  @GIN(B322);  @GIN(B331);  @GIN(B332);  @GIN(B341);
@GIN(B342);  @GIN(B411);  @GIN(B412);  @GIN(B421);  @GIN(B422);  @GIN(B431);  @GIN(B432);

!Declare the X Variables as the Binary Integer Variables;
@BIN(X111);  @BIN(X112);  @BIN(X113);  @BIN(X121);  @BIN(X122);  @BIN(X123);
@BIN(X211);  @BIN(X212);  @BIN(X213);  @BIN(X221);  @BIN(X222);  @BIN(X223);
@BIN(X311);  @BIN(X312);  @BIN(X313);
@BIN(X411);  @BIN(X412);  @BIN(X413);  @BIN(X421);  @BIN(X422);  @BIN(X423);
@BIN(X511);  @BIN(X512);  @BIN(X513);

END
!End of MODEL;
```

APPENDIX A.2

First Transformed Model of the First Problem Instance

Note: This mathematical model is the result of applying the first transformation technique to the original model presented in Appendix A.1. The technique transforms all the general integer variables into binary variables. As a result, the model is a pure binary integer non-linear programming model. In the original model, the general integer variables are represented by variable B(j,k,rt). They are bounded between 0 and 3. Upon transformation, variable B(j,k,rt) becomes variable B(j,k,rt,b), where the additional term 'b' is added for the transformation purpose.

```
! OBJECTIVE FUNCTION;

MODEL:

MIN =

!Part 1 – Operation 1;
30*S1111*B1111 + 60*S1111*B1112 +
46*S1112*B1111 + 92*S1112*B1112 + 58*S1112*B1121 + 116*S1112*B1122 +
21*S1113*B1111 + 42*S1113*B1112 + 30*S1121*B1111 + 60*S1121*B1112 +
46*S1122*B1111 + 92*S1122*B1112 + 58*S1122*B1121 + 116*S1122*B1122 +
21*S1123*B1111 + 42*S1123*B1112 +
!Part 1 – Operation 2;
39*S1111*S1212*B1211 + 78*S1111*S1212*B1212 +
64*S1111*S1213*B1211 + 128*S1111*S1213*B1212 + 76*S1111*S1213*B1221 + 152*S1111*S1213*B1222 +
33*S1112*S1211*B1211 + 66*S1112*S1211*B1212 +
52*S1112*S1213*B1211 + 104*S1112*S1213*B1212 + 69*S1112*S1213*B1221 + 138*S1112*S1213*B1222 +
43*S1113*S1211*B1211 + 86*S1113*S1211*B1212 + 70*S1113*S1211*B1221 + 140*S1113*S1211*B1222 +
52*S1113*S1212*B1211 + 104*S1113*S1212*B1212 + 54*S1113*S1212*B1221 + 108*S1113*S1212*B1222 +
39*S1121*S1212*B1211 + 78*S1121*S1212*B1212 +
64*S1121*S1213*B1211 + 128*S1121*S1213*B1212 + 76*S1121*S1213*B1221 + 152*S1121*S1213*B1222 +
33*S1122*S1211*B1211 + 66*S1122*S1211*B1212 +
52*S1122*S1213*B1211 + 104*S1122*S1213*B1212 + 69*S1122*S1213*B1221 + 138*S1122*S1213*B1222 +
43*S1123*S1211*B1211 + 86*S1123*S1211*B1212 + 70*S1123*S1211*B1221 + 140*S1123*S1211*B1222 +
52*S1123*S1212*B1211 + 104*S1123*S1212*B1212 + 54*S1123*S1212*B1221 + 108*S1123*S1212*B1222 +
!Part 1 – Operation 3;
39*S1211*S1312*B1311 + 78*S1211*S1312*B1312 +
64*S1211*S1313*B1311 + 128*S1211*S1313*B1312 + 76*S1211*S1313*B1321 + 152*S1211*S1313*B1322 +
33*S1212*S1311*B1311 + 66*S1212*S1311*B1312 +
52*S1212*S1313*B1311 + 104*S1212*S1313*B1312 + 69*S1212*S1313*B1321 + 138*S1212*S1313*B1322 +
43*S1213*S1311*B1311 + 86*S1213*S1311*B1312 + 70*S1213*S1311*B1321 + 140*S1213*S1311*B1322 +
52*S1213*S1312*B1311 + 104*S1213*S1312*B1312 + 54*S1213*S1312*B1321 + 108*S1213*S1312*B1322 +
39*S1211*S1322*B1311 + 78*S1211*S1322*B1312 +
64*S1211*S1323*B1311 + 128*S1211*S1323*B1312 + 76*S1211*S1323*B1321 + 152*S1211*S1323*B1322 +
33*S1212*S1321*B1311 + 66*S1212*S1321*B1312 +
52*S1212*S1323*B1311 + 104*S1212*S1323*B1312 + 69*S1212*S1323*B1321 + 138*S1212*S1323*B1322 +
43*S1213*S1321*B1311 + 86*S1213*S1321*B1312 + 70*S1213*S1321*B1321 + 140*S1213*S1321*B1322 +
52*S1213*S1322*B1311 + 104*S1213*S1322*B1312 + 54*S1213*S1322*B1321 + 108*S1213*S1322*B1322 +
!Part 1 – Operation 4;
48*S1311*B1411 + 96*S1311*B1412 +
55*S1312*B1411 + 110*S1312*B1412 + 58*S1312*B1421 + 116*S1312*B1422 +
18*S1313*B1411 + 36*S1313*B1412 + 48*S1321*B1411 + 96*S1321*B1412 +
55*S1322*B1411 + 110*S1322*B1412 + 58*S1322*B1421 + 116*S1322*B1422 +
18*S1323*B1411 + 36*S1323*B1412 +
!Part 2 – Operation 1;
30*S2111*B2111 + 60*S2111*B2112 +
46*S2112*B2111 + 92*S2112*B2112 + 58*S2112*B2121 + 116*S2112*B2122 +
21*S2113*B2111 + 42*S2113*B2112 + 30*S2121*B2111 + 60*S2121*B2112 +
```

```
46*S2122*B2111 + 92*S2122*B2112 + 58*S2122*B2121 + 116*S2122*B2122 +
21*S2123*B2111 + 42*S2123*B2112 +
!Part 2 - Operation 2;
39*S2111*S2212*B2211 + 78*S2111*S2212*B2212 +
64*S2111*S2213*B2211 + 128*S2111*S2213*B2212 + 76*S2111*S2213*B2221 + 152*S2111*S2213*B2222 +
33*S2112*S2211*B2211 + 66*S2112*S2211*B2212 +
52*S2112*S2213*B2211 + 104*S2112*S2213*B2212 + 69*S2112*S2213*B2221 + 138*S2112*S2213*B2222 +
43*S2113*S2211*B2211 + 86*S2113*S2211*B2212 + 70*S2113*S2211*B2221 +
140*S2113*S2211*B2222 +
52*S2113*S2212*B2211 + 104*S2113*S2212*B2212 + 54*S2113*S2212*B2221 + 108*S2113*S2212*B2222 +
39*S2111*S2222*B2211 + 78*S2111*S2222*B2212 +
64*S2111*S2223*B2211 + 128*S2111*S2223*B2212 + 76*S2111*S2223*B2221 + 152*S2111*S2223*B2222 +
33*S2112*S2221*B2211 + 66*S2112*S2221*B2212 +
52*S2112*S2223*B2211 + 104*S2112*S2223*B2212 + 69*S2112*S2223*B2221 + 138*S2112*S2223*B2222 +
43*S2113*S2221*B2211 + 86*S2113*S2221*B2212 + 70*S2113*S2221*B2221 + 140*S2113*S2221*B2222 +
52*S2113*S2222*B2211 + 104*S2113*S2222*B2212 + 54*S2113*S2222*B2221 + 108*S2113*S2222*B2222 +
39*S2121*S2212*B2211 + 78*S2121*S2212*B2212 +
64*S2121*S2213*B2211 + 128*S2121*S2213*B2212 + 76*S2121*S2213*B2221 + 152*S2121*S2213*B2222 +
33*S2122*S2211*B2211 + 66*S2122*S2211*B2212 +
52*S2122*S2213*B2211 + 104*S2122*S2213*B2212 + 69*S2122*S2213*B2221 + 138*S2122*S2213*B2222 +
43*S2123*S2211*B2211 + 86*S2123*S2211*B2212 + 70*S2123*S2211*B2221 + 140*S2123*S2211*B2222 +
52*S2123*S2212*B2211 + 104*S2123*S2212*B2212 + 54*S2123*S2212*B2221 + 108*S2123*S2212*B2222 +
39*S2121*S2222*B2211 + 78*S2121*S2222*B2212 +
64*S2121*S2223*B2211 + 128*S2121*S2223*B2212 + 76*S2121*S2223*B2221 + 152*S2121*S2223*B2222 +
33*S2122*S2221*B2211 + 66*S2122*S2221*B2212 +
52*S2122*S2223*B2211 + 104*S2122*S2223*B2212 + 69*S2122*S2223*B2221 + 138*S2122*S2223*B2222 +
43*S2123*S2221*B2211 + 86*S2123*S2221*B2212 + 70*S2123*S2221*B2221 + 140*S2123*S2221*B2222 +
52*S2123*S2222*B2211 + 104*S2123*S2222*B2212 + 54*S2123*S2222*B2221 + 108*S2123*S2222*B2222 +
!Part 2 – Operation 3;
48*S2211*B2311 + 96*S2211*B2312 +
55*S2212*B2311 + 110*S2212*B2312 + 58*S2212*B2321 + 116*S2212*B2322 +
18*S2213*B2311 + 36*S2213*B2312 + 48*S2221*B2311 + 96*S2221*B2312 +
55*S2222*B2311 + 110*S2222*B2312 + 58*S2222*B2321 + 116*S2222*B2322 +
18*S2223*B2311 + 36*S2223*B2312 +
!Part 3 – Operation 1;
30*S3111*B3111 + 60*S3111*B3112 +
46*S3112*B3111 + 92*S3112*B3112 + 58*S3112*B3121 + 116*S3112*B3122 +
21*S3113*B3111 + 42*S3113*B3112 + 30*S3121*B3111 + 60*S3121*B3112 +
46*S3122*B3111 + 92*S3122*B3112 + 58*S3122*B3121 + 116*S3122*B3122 +
21*S3123*B3111 + 42*S3123*B3112 +
!Part 3 - Operation 2;
39*S3111*S3212*B3211 + 78*S3111*S3212*B3212 +
64*S3111*S3213*B3211 + 128*S3111*S3213*B3212 + 76*S3111*S3213*B3221 + 152*S3111*S3213*B3222 +
33*S3112*S3211*B3211 + 66*S3112*S3211*B3212 +
52*S3112*S3213*B3211 + 104*S3112*S3213*B3212 + 69*S3112*S3213*B3221 + 138*S3112*S3213*B3222 +
43*S3113*S3211*B3211 + 86*S3113*S3211*B3212 + 70*S3113*S3211*B3221 + 140*S3113*S3211*B3222 +
52*S3113*S3212*B3211 + 104*S3113*S3212*B3212 + 54*S3113*S3212*B3221 + 108*S3113*S3212*B3222 +
39*S3121*S3212*B3211 + 78*S3121*S3212*B3212 +
64*S3121*S3213*B3211 + 128*S3121*S3213*B3212 + 76*S3121*S3213*B3221 + 152*S3121*S3213*B3222 +
33*S3122*S3211*B3211 + 66*S3122*S3211*B3212 +
52*S3122*S3213*B3211 + 104*S3122*S3213*B3212 + 69*S3122*S3213*B3221 + 138*S3122*S3213*B3222 +
43*S3123*S3211*B3211 + 86*S3123*S3211*B3212 + 70*S3123*S3211*B3221 + 140*S3123*S3211*B3222 +
52*S3123*S3212*B3211 + 104*S3123*S3212*B3212 + 54*S3123*S3212*B3221 + 108*S3123*S3212*B3222 +
!Part 3 – Operation 3;
39*S3211*S3312*B3311 + 78*S3211*S3312*B3312 +
64*S3211*S3313*B3311 + 128*S3211*S3313*B3312 + 76*S3211*S3313*B3321 + 152*S3211*S3313*B3322 +
33*S3212*S3311*B3311 + 66*S3212*S3311*B3312 +
52*S3212*S3313*B3311 + 104*S3212*S3313*B3312 + 69*S3212*S3313*B3321 + 138*S3212*S3313*B3322 +
43*S3213*S3311*B3311 + 86*S3213*S3311*B3312 + 70*S3213*S3311*B3321 + 140*S3213*S3311*B3322 +
52*S3213*S3312*B3311 + 104*S3213*S3312*B3312 + 54*S3213*S3312*B3321 + 108*S3213*S3312*B3322 +
39*S3211*S3322*B3311 + 78*S3211*S3322*B3312 +
64*S3211*S3323*B3311 + 128*S3211*S3323*B3312 + 76*S3211*S3323*B3321 + 152*S3211*S3323*B3322 +
33*S3212*S3321*B3311 + 66*S3212*S3321*B3312 +
52*S3212*S3323*B3311 + 104*S3212*S3323*B3312 + 69*S3212*S3323*B3321 + 138*S3212*S3323*B3322 +
43*S3213*S3321*B3311 + 86*S3213*S3321*B3312 + 70*S3213*S3321*B3321 + 140*S3213*S3321*B3322 +
52*S3213*S3322*B3311 + 104*S3213*S3322*B3312 + 54*S3213*S3322*B3321 + 108*S3213*S3322*B3322 +
!Part 3 – Operation 4;
48*S3311*B3411 + 96*S3311*B3412 +
55*S3312*B3411 + 110*S3312*B3412 + 58*S3312*B3421 + 116*S3312*B3422 +
18*S3313*B3411 + 36*S3313*B3412 + 48*S3321*B3411 + 96*S3321*B3412 +
```

```
55*S3322*B3411 + 110*S3322*B3412 + 58*S3322*B3421 + 116*S3322*B3422 +
18*S3323*B3411 + 36*S3323*B3412 +
!Part 4 – Operation 1;
30*S4111*B4111 + 60*S4111*B4112 +
46*S4112*B4111 + 92*S4112*B4112 + 58*S4112*B4121 + 116*S4112*B4122 +
21*S4113*B4111 + 42*S4113*B4112 + 30*S4121*B4111 + 60*S4121*B4112 +
46*S4122*B4111 + 92*S4122*B4112 + 58*S4122*B4121 + 116*S4122*B4122 +
21*S4123*B4111 + 42*S4123*B4112 +
!Part 4 – Operation 2;
39*S4111*S4212*B4211 + 78*S4111*S4212*B4212 +
64*S4111*S4213*B4211 + 128*S4111*S4213*B4212 + 76*S4111*S4213*B4221 + 152*S4111*S4213*B4222 +
33*S4112*S4211*B4211 + 66*S4112*S4211*B4212 +
52*S4112*S4213*B4211 + 104*S4112*S4213*B4212 + 69*S4112*S4213*B4221 + 138*S4112*S4213*B4222 +
43*S4113*S4211*B4211 + 86*S4113*S4211*B4212 + 70*S4113*S4211*B4221 + 140*S4113*S4211*B4222 +
52*S4113*S4212*B4211 + 104*S4113*S4212*B4212 + 54*S4113*S4212*B4221 + 108*S4113*S4212*B4222 +
39*S4121*S4212*B4211 + 78*S4121*S4212*B4212 +
64*S4121*S4213*B4211 + 128*S4121*S4213*B4212 + 76*S4121*S4213*B4221 + 152*S4121*S4213*B4222 +
33*S4122*S4211*B4211 + 66*S4122*S4211*B4212 +
52*S4122*S4213*B4211 + 104*S4122*S4213*B4212 + 69*S4122*S4213*B4221 + 138*S4122*S4213*B4222 +
43*S4123*S4211*B4211 + 86*S4123*S4211*B4212 + 70*S4123*S4211*B4221 + 140*S4123*S4211*B4222 +
52*S4123*S4212*B4211 + 104*S4123*S4212*B4212 + 54*S4123*S4212*B4221 + 108*S4123*S4212*B4222 +
!Part 4 – Operation 3;
48*S4211*B4311 + 96*S4211*B4312 +
55*S4212*B4311 + 110*S4212*B4312 + 58*S4212*B4321 + 116*S4212*B4322 +
18*S4213*B4311 + 36*S4213*B4312;

!End of OBJECTIVE FUNCTION;


!OPERATIONAL CONSTRAINTS;

!First Constraint - No Split Operations;
S1111+ S1112+ S1113+ S1121+ S1122+ S1123 = 1;
S1211+ S1212+ S1213 = 1;
S1311+ S1312+ S1313+ S1321+ S1322+ S1323 = 1;
S2111+ S2112+ S2113+ S2121+ S2122+ S2123 = 1;
S2211+ S2212+ S2213+ S2221+ S2222+ S2223 = 1;
S3111+ S3112+ S3113+ S3121+ S3122+ S3123 = 1;
S3211+ S3212+ S3213 = 1;
S3311+ S3312+ S3313+ S3321+ S3322+ S3323 = 1;
S4111+ S4112+ S4113+ S4121+ S4122+ S4123 = 1;
S4211+ S4212+ S4213 = 1;

!Second Constraint - Only One Unique Machine In Each Location;
X111+ X112+ X113 = 1;
X121+ X122+ X123 = 1;
X211+ X212+ X213 = 1;
X221+ X222+ X223 = 1;
X311+ X312+ X313 = 1;
X411+ X412+ X413 = 1;
X421+ X422+ X423 = 1;
X511+ X512+ X513 = 1;

!Third Constraint - Machine Capacity Limitations;
8*X111 - 4.0*S1111 - 7.0*S4111 >= 0;
8*X112 - 4.0*S1112 - 7.0*S4112 >= 0;
8*X113 - 4.0*S1113 - 7.0*S4113 >= 0;
8*X121 - 4.0*S1121 - 7.0*S4121 >= 0;
8*X122 - 4.0*S1122 - 7.0*S4122 >= 0;
8*X123 - 4.0*S1123 - 7.0*S4123 >= 0;
8*X211 - 3.5*S2111 - 4.0*S3111 - 6.0*S3311 >= 0;
8*X212 - 3.5*S2112 - 4.0*S3112 - 6.0*S3312 >= 0;
8*X213 - 3.5*S2113 - 4.0*S3113 - 6.0*S3313 >= 0;
8*X221 - 3.5*S2121 - 4.0*S3121 - 6.0*S3321 >= 0;
8*X222 - 3.5*S2122 - 4.0*S3122 - 6.0*S3322 >= 0;
8*X223 - 3.5*S2123 - 4.0*S3123 - 6.0*S3323 >= 0;
8*X311 - 5.0*S1211 - 3.0*S4211 >= 0;
8*X312 - 5.0*S1212 - 3.0*S4212 >= 0;
8*X313 - 5.0*S1213 - 3.0*S4213 >= 0;
8*X411 - 6.0*S1311 - 4.5*S2211 >= 0;
```

```
8*X412 - 6.0*S1312 - 4.5*S2212 >= 0;
8*X413 - 6.0*S1313 - 4.5*S2213 >= 0;
8*X421 - 6.0*S1321 - 4.5*S2221 >= 0;
8*X422 - 6.0*S1322 - 4.5*S2222 >= 0;
8*X423 - 6.0*S1323 - 4.5*S2223 >= 0;
8*X511 - 5.0*S3211 >= 0;
8*X512 - 5.0*S3212 >= 0;
8*X513 - 5.0*S3213 >= 0;

!Fourth Constraint - Maximum Number of Machines in a Location;
X111+ X121+ X211+ X221+ X311+ X411+ X421+ X511 <= 3;
X112+ X122+ X212+ X222+ X312+ X412+ X422+ X512 <= 3;
X113+ X123+ X213+ X223+ X313+ X413+ X423+ X513 <= 3;

!Fifth Constraint - Demand for each part;
!Part 1 - Operation 1;
1*S1111*B1111+ 2*S1111*B1112+
1*S1112*B1111+ 2*S1112*B1112+ 1*S1112*B1121+ 2*S1112*B1122+
1*S1113*B1111+ 2*S1113*B1112+ 1*S1121*B1111+ 2*S1121*B1112+
1*S1122*B1111+ 2*S1122*B1112+ 1*S1122*B1121+ 2*S1122*B1122+
1*S1123*B1111+ 2*S1123*B1112 = 3;
!Part 1 - Operation 2;
1*S1111*S1211*B1211+ 2*S1111*S1211*B1212+ 1*S1111*S1212*B1211+ 2*S1111*S1212*B1212+
1*S1111*S1213*B1211+ 2*S1111*S1213*B1212+ 1*S1111*S1213*B1221+ 2*S1111*S1213*B1222+
1*S1112*S1211*B1211+ 2*S1112*S1211*B1212+ 1*S1112*S1212*B1211+ 2*S1112*S1212*B1212+
1*S1112*S1213*B1211+ 2*S1112*S1213*B1212+ 1*S1112*S1213*B1221+ 2*S1112*S1213*B1222+
1*S1113*S1211*B1211+ 2*S1113*S1211*B1212+ 1*S1113*S1211*B1221+ 2*S1113*S1211*B1222+
1*S1113*S1212*B1211+ 2*S1113*S1212*B1212+ 1*S1113*S1212*B1221+ 2*S1113*S1212*B1222+
1*S1113*S1213*B1211+ 2*S1113*S1213*B1212+ 1*S1121*S1211*B1211+ 2*S1121*S1211*B1212+
1*S1121*S1212*B1211+ 2*S1121*S1212*B1212+
1*S1121*S1213*B1211+ 2*S1121*S1213*B1212+ 1*S1121*S1213*B1221+ 2*S1121*S1213*B1222+
1*S1122*S1211*B1211+ 2*S1122*S1211*B1212+ 1*S1122*S1212*B1211+ 2*S1122*S1212*B1212+
1*S1122*S1213*B1211+ 2*S1122*S1213*B1212+ 1*S1122*S1213*B1221+ 2*S1122*S1213*B1222+
1*S1123*S1211*B1211+ 2*S1123*S1211*B1212+ 1*S1123*S1211*B1221+ 2*S1123*S1211*B1222+
1*S1123*S1212*B1211+ 2*S1123*S1212*B1212+ 1*S1123*S1212*B1221+ 2*S1123*S1212*B1222+
1*S1123*S1213*B1211+ 2*S1123*S1213*B1212 = 3;
!Part 1 - Operation 3;
1*S1211*S1311*B1311+ 2*S1211*S1311*B1312+ 1*S1211*S1312*B1311+ 2*S1211*S1312*B1312+
1*S1211*S1313*B1311+ 2*S1211*S1313*B1312+ 1*S1211*S1313*B1321+ 2*S1211*S1313*B1322+
1*S1212*S1311*B1311+ 2*S1212*S1311*B1312+ 1*S1212*S1312*B1311+ 2*S1212*S1312*B1312+
1*S1212*S1313*B1311+ 2*S1212*S1313*B1312+ 1*S1212*S1313*B1321+ 2*S1212*S1313*B1322+
1*S1213*S1311*B1311+ 2*S1213*S1311*B1312+ 1*S1213*S1311*B1321+ 2*S1213*S1311*B1322+
1*S1213*S1312*B1311+ 2*S1213*S1312*B1312+ 1*S1213*S1312*B1321+ 2*S1213*S1312*B1322+
1*S1213*S1313*B1311+ 2*S1213*S1313*B1312+ 1*S1211*S1321*B1311+ 2*S1211*S1321*B1312+
1*S1211*S1322*B1311+ 2*S1211*S1322*B1312+
1*S1211*S1323*B1311+ 2*S1211*S1323*B1312+ 1*S1211*S1323*B1321+ 2*S1211*S1323*B1322+
1*S1212*S1321*B1311+ 2*S1212*S1321*B1312+ 1*S1212*S1322*B1311+ 2*S1212*S1322*B1312+
1*S1212*S1323*B1311+ 2*S1212*S1323*B1312+ 1*S1212*S1323*B1321+ 2*S1212*S1323*B1322+
1*S1213*S1321*B1311+ 2*S1213*S1321*B1312+ 1*S1213*S1321*B1321+ 2*S1213*S1321*B1322+
1*S1213*S1322*B1311+ 2*S1213*S1322*B1312+ 1*S1213*S1322*B1321+ 2*S1213*S1322*B1322+
1*S1213*S1323*B1311+ 2*S1213*S1323*B1312 = 3;
!Part 1 - Operation 4;
1*S1311*B1411+ 2*S1311*B1412+
1*S1312*B1411+ 2*S1312*B1412+ 1*S1312*B1421+ 2*S1312*B1422+
1*S1313*B1411+ 2*S1313*B1412+ 1*S1321*B1411+ 2*S1321*B1412+
1*S1322*B1411+ 2*S1322*B1412+ 1*S1322*B1421+ 2*S1322*B1422+
1*S1323*B1411+ 2*S1323*B1412 = 3;
!Part 2 - Operation 1;
1*S2111*B2111+ 2*S2111*B2112+
1*S2112*B2111+ 2*S2112*B2112+ 1*S2112*B2121+ 2*S2112*B2122+
1*S2113*B2111+ 2*S2113*B2112+ 1*S2121*B2111+ 2*S2121*B2112+
1*S2122*B2111+ 2*S2122*B2112+ 1*S2122*B2121+ 2*S2122*B2122+
1*S2123*B2111+ 2*S2123*B2112 = 2;
!Part 2 - Operation 2;
1*S2111*S2211*B2211+ 2*S2111*S2211*B2212+ 1*S2111*S2212*B2211+ 2*S2111*S2212*B2212+
1*S2111*S2213*B2211+ 2*S2111*S2213*B2212+ 1*S2111*S2213*B2221+ 2*S2111*S2213*B2222+
1*S2112*S2211*B2211+ 2*S2112*S2211*B2212+ 1*S2112*S2212*B2211+ 2*S2112*S2212*B2212+
1*S2112*S2213*B2211+ 2*S2112*S2213*B2212+ 1*S2112*S2213*B2221+ 2*S2112*S2213*B2222+
1*S2113*S2211*B2211+ 2*S2113*S2211*B2212+ 1*S2113*S2211*B2221+ 2*S2113*S2211*B2222+
1*S2113*S2212*B2211+ 2*S2113*S2212*B2212+ 1*S2113*S2212*B2221+ 2*S2113*S2212*B2222+
```

```
1*S2113*S2213*B2211+ 2*S2113*S2213*B2212+ 1*S2111*S2221*B2211+ 2*S2111*S2221*B2212+
1*S2111*S2222*B2211+ 2*S2111*S2222*B2212+
1*S2111*S2223*B2211+ 2*S2111*S2223*B2212+ 1*S2111*S2223*B2221+ 2*S2111*S2223*B2222+
1*S2112*S2221*B2211+ 2*S2112*S2221*B2212+ 1*S2112*S2222*B2211+ 2*S2112*S2222*B2212+
1*S2112*S2223*B2211+ 2*S2112*S2223*B2212+ 1*S2112*S2223*B2221+ 2*S2112*S2223*B2222+
1*S2113*S2221*B2211+ 2*S2113*S2221*B2212+ 1*S2113*S2221*B2221+ 2*S2113*S2221*B2222+
1*S2113*S2222*B2211+ 2*S2113*S2222*B2212+ 1*S2113*S2222*B2221+ 2*S2113*S2222*B2222+
1*S2113*S2223*B2211+ 2*S2113*S2223*B2212+ 1*S2121*S2211*B2211+ 2*S2121*S2211*B2212+
1*S2121*S2212*B2211+ 2*S2121*S2212*B2212+
1*S2121*S2213*B2211+ 2*S2121*S2213*B2212+ 1*S2121*S2213*B2221+ 2*S2121*S2213*B2222+
1*S2122*S2211*B2211+ 2*S2122*S2211*B2212+ 1*S2122*S2212*B2211+ 2*S2122*S2212*B2212+
1*S2122*S2213*B2211+ 2*S2122*S2213*B2212+ 1*S2122*S2213*B2221+ 2*S2122*S2213*B2222+
1*S2123*S2211*B2211+ 2*S2123*S2211*B2212+ 1*S2123*S2211*B2221+ 2*S2123*S2211*B2222+
1*S2123*S2212*B2211+ 2*S2123*S2212*B2212+ 1*S2123*S2212*B2221+ 2*S2123*S2212*B2222+
1*S2123*S2213*B2211+ 2*S2123*S2213*B2212+ 1*S2121*S2221*B2211+ 2*S2121*S2221*B2212+
1*S2121*S2222*B2211+ 2*S2121*S2222*B2212+
1*S2121*S2223*B2211+ 2*S2121*S2223*B2212+ 1*S2121*S2223*B2221+ 2*S2121*S2223*B2222+
1*S2122*S2221*B2211+ 2*S2122*S2221*B2212+ 1*S2122*S2222*B2211+ 2*S2122*S2222*B2212+
1*S2122*S2223*B2211+ 2*S2122*S2223*B2212+ 1*S2122*S2223*B2221+ 2*S2122*S2223*B2222+
1*S2123*S2221*B2211+ 2*S2123*S2221*B2212+ 1*S2123*S2221*B2221+ 2*S2123*S2221*B2222+
1*S2123*S2222*B2211+ 2*S2123*S2222*B2212+ 1*S2123*S2222*B2221+ 2*S2123*S2222*B2222+
1*S2123*S2223*B2211+ 2*S2123*S2223*B2212 = 2;
!Part 2 - Operation 3;
1*S2211*B2311+ 2*S2211*B2312+
1*S2212*B2311+ 2*S2212*B2312+ 1*S2212*B2321+ 2*S2212*B2322+
1*S2213*B2311+ 2*S2213*B2312+ 1*S2221*B2311+ 2*S2221*B2312+
1*S2222*B2311+ 2*S2222*B2312+ 1*S2222*B2321+ 2*S2222*B2322+
1*S2223*B2311+ 2*S2223*B2312 = 2;
!Part 3 - Operation 1;
1*S3111*B3111+ 2*S3111*B3112+
1*S3112*B3111+ 2*S3112*B3112+ 1*S3112*B3121+ 2*S3112*B3122+
1*S3113*B3111+ 2*S3113*B3112+ 1*S3121*B3111+ 2*S3121*B3112+
1*S3122*B3111+ 2*S3122*B3112+ 1*S3122*B3121+ 2*S3122*B3122+
1*S3123*B3111+ 2*S3123*B3112 = 3;
!Part 3 - Operation 2;
1*S3111*S3211*B3211+ 2*S3111*S3211*B3212+ 1*S3111*S3212*B3211+ 2*S3111*S3212*B3212+
1*S3111*S3213*B3211+ 2*S3111*S3213*B3212+ 1*S3111*S3213*B3221+ 2*S3111*S3213*B3222+
1*S3112*S3211*B3211+ 2*S3112*S3211*B3212+ 1*S3112*S3212*B3211+ 2*S3112*S3212*B3212+
1*S3112*S3213*B3211+ 2*S3112*S3213*B3212+ 1*S3112*S3213*B3221+ 2*S3112*S3213*B3222+
1*S3113*S3211*B3211+ 2*S3113*S3211*B3212+ 1*S3113*S3211*B3221+ 2*S3113*S3211*B3222+
1*S3113*S3212*B3211+ 2*S3113*S3212*B3212+ 1*S3113*S3212*B3221+ 2*S3113*S3212*B3222+
1*S3113*S3213*B3211+ 2*S3113*S3213*B3212+ 1*S3121*S3211*B3211+ 2*S3121*S3211*B3212+
1*S3121*S3212*B3211+ 2*S3121*S3212*B3212+
1*S3121*S3213*B3211+ 2*S3121*S3213*B3212+ 1*S3121*S3213*B3221+ 2*S3121*S3213*B3222+
1*S3122*S3211*B3211+ 2*S3122*S3211*B3212+ 1*S3122*S3212*B3211+ 2*S3122*S3212*B3212+
1*S3122*S3213*B3211+ 2*S3122*S3213*B3212+ 1*S3122*S3213*B3221+ 2*S3122*S3213*B3222+
1*S3123*S3211*B3211+ 2*S3123*S3211*B3212+ 1*S3123*S3211*B3221+ 2*S3123*S3211*B3222+
1*S3123*S3212*B3211+ 2*S3123*S3212*B3212+ 1*S3123*S3212*B3221+ 2*S3123*S3212*B3222+
1*S3123*S3213*B3211+ 2*S3123*S3213*B3212 = 3;
!Part 3 - Operation 3;
1*S3211*S3311*B3311+ 2*S3211*S3311*B3312+ 1*S3211*S3312*B3311+ 2*S3211*S3312*B3312+
1*S3211*S3313*B3311+ 2*S3211*S3313*B3312+ 1*S3211*S3313*B3321+ 2*S3211*S3313*B3322+
1*S3212*S3311*B3311+ 2*S3212*S3311*B3312+ 1*S3212*S3312*B3311+ 2*S3212*S3312*B3312+
1*S3212*S3313*B3311+ 2*S3212*S3313*B3312+ 1*S3212*S3313*B3321+ 2*S3212*S3313*B3322+
1*S3213*S3311*B3311+ 2*S3213*S3311*B3312+ 1*S3213*S3311*B3321+ 2*S3213*S3311*B3322+
1*S3213*S3312*B3311+ 2*S3213*S3312*B3312+ 1*S3213*S3312*B3321+ 2*S3213*S3312*B3322+
1*S3213*S3313*B3311+ 2*S3213*S3313*B3312+ 1*S3211*S3321*B3311+ 2*S3211*S3321*B3312+
1*S3211*S3322*B3311+ 2*S3211*S3322*B3312+
1*S3211*S3323*B3311+ 2*S3211*S3323*B3312+ 1*S3211*S3323*B3321+ 2*S3211*S3323*B3322+
1*S3212*S3321*B3311+ 2*S3212*S3321*B3312+ 1*S3212*S3322*B3311+ 2*S3212*S3322*B3312+
1*S3212*S3323*B3311+ 2*S3212*S3323*B3312+ 1*S3212*S3323*B3321+ 2*S3212*S3323*B3322+
1*S3213*S3321*B3311+ 2*S3213*S3321*B3312+ 1*S3213*S3321*B3321+ 2*S3213*S3321*B3322+
1*S3213*S3322*B3311+ 2*S3213*S3322*B3312+ 1*S3213*S3322*B3321+ 2*S3213*S3322*B3322+
1*S3213*S3323*B3311+ 2*S3213*S3323*B3312 = 3;
!Part 3 - Operation 4;
1*S3311*B3411+ 2*S3311*B3412+
1*S3312*B3411+ 2*S3312*B3412+ 1*S3312*B3421+ 2*S3312*B3422+
1*S3313*B3411+ 2*S3313*B3412+ 1*S3321*B3411+ 2*S3321*B3412+
1*S3322*B3411+ 2*S3322*B3412+ 1*S3322*B3421+ 2*S3322*B3422+
1*S3323*B3411+ 2*S3323*B3412 = 3;
```

```
!Part 4 - Operation 1;
1*S4111*B4111+ 2*S4111*B4112+
1*S4112*B4111+ 2*S4112*B4112+ 1*S4112*B4121+ 2*S4112*B4122+
1*S4113*B4111+ 2*S4113*B4112+ 1*S4121*B4111+ 2*S4121*B4112+
1*S4122*B4111+ 2*S4122*B4112+ 1*S4122*B4121+ 2*S4122*B4122+
1*S4123*B4111+ 2*S4123*B4112 = 2;
!Part 4 - Operation 2;
1*S4111*S4211*B4211+ 2*S4111*S4211*B4212+ 1*S4111*S4212*B4211+ 2*S4111*S4212*B4212+
1*S4111*S4213*B4211+ 2*S4111*S4213*B4212+ 1*S4111*S4213*B4221+ 2*S4111*S4213*B4222+
1*S4112*S4211*B4211+ 2*S4112*S4211*B4212+ 1*S4112*S4212*B4211+ 2*S4112*S4212*B4212+
1*S4112*S4213*B4211+ 2*S4112*S4213*B4212+ 1*S4112*S4213*B4221+ 2*S4112*S4213*B4222+
1*S4113*S4211*B4211+ 2*S4113*S4211*B4212+ 1*S4113*S4211*B4221+ 2*S4113*S4211*B4222+
1*S4113*S4212*B4211+ 2*S4113*S4212*B4212+ 1*S4113*S4212*B4221+ 2*S4113*S4212*B4222+
1*S4113*S4213*B4211+ 2*S4113*S4213*B4212+ 1*S4121*S4211*B4211+ 2*S4121*S4211*B4212+
1*S4121*S4212*B4211+ 2*S4121*S4212*B4212+
1*S4121*S4213*B4211+ 2*S4121*S4213*B4212+ 1*S4121*S4213*B4221+ 2*S4121*S4213*B4222+
1*S4122*S4211*B4211+ 2*S4122*S4211*B4212+ 1*S4122*S4212*B4211+ 2*S4122*S4212*B4212+
1*S4122*S4213*B4211+ 2*S4122*S4213*B4212+ 1*S4122*S4213*B4221+ 2*S4122*S4213*B4222+
1*S4123*S4211*B4211+ 2*S4123*S4211*B4212+ 1*S4123*S4211*B4221+ 2*S4123*S4211*B4222+
1*S4123*S4212*B4211+ 2*S4123*S4212*B4212+ 1*S4123*S4212*B4221+ 2*S4123*S4212*B4222+
1*S4123*S4213*B4211+ 2*S4123*S4213*B4212 = 2;
!Part 4 - Operation 3;
1*S4211*B4311+ 2*S4211*B4312+
1*S4212*B4311+ 2*S4212*B4312+ 1*S4212*B4321+ 2*S4212*B4322+
1*S4213*B4311+ 2*S4213*B4312 = 2;

!Sixth Constraint - Demand for each AGV;

!AGV 1;
48*S1111*B1111+ 96*S1111*B1112+ 48*S1112*B1111+ 96*S1112*B1112+
8*S1112*B1121+ 16*S1112*B1122+ 8*S1113*B1111+ 16*S1113*B1112+
48*S1121*B1111+ 96*S1121*B1112+ 48*S1122*B1111+ 96*S1122*B1112+
8*S1122*B1121+ 16*S1122*B1122+ 8*S1123*B1111+ 16*S1123*B1112+
48*S2111*B2111+ 96*S2111*B2112+ 48*S2112*B2111+ 96*S2112*B2112+
8*S2112*B2121+ 16*S2112*B2122+ 8*S2113*B2111+ 16*S2113*B2112+
48*S2121*B2111+ 96*S2121*B2112+ 48*S2122*B2111+ 96*S2122*B2112+
8*S2122*B2121+ 16*S2122*B2122+ 8*S2123*B2111+ 16*S2123*B2112+
48*S3111*B3111+ 96*S3111*B3112+ 48*S3112*B3111+ 96*S3112*B3112+
8*S3112*B3121+ 16*S3112*B3122+ 8*S3113*B3111+ 16*S3113*B3112+
48*S3121*B3111+ 96*S3121*B3112+ 48*S3122*B3111+ 96*S3122*B3112+
8*S3122*B3121+ 16*S3122*B3122+ 8*S3123*B3111+ 16*S3123*B3112+
48*S4111*B4111+ 96*S4111*B4112+ 48*S4112*B4111+ 96*S4112*B4112+
8*S4112*B4121+ 16*S4112*B4122+ 8*S4113*B4111+ 16*S4113*B4112+
48*S4121*B4111+ 96*S4121*B4112+ 48*S4122*B4111+ 96*S4122*B4112+
8*S4122*B4121+ 16*S4122*B4122+ 8*S4123*B4111+ 16*S4123*B4112+
48*S1311*B1411+ 96*S1311*B1412+ 8*S1312*B1411+ 16*S1312*B1412+
48*S1312*B1421+ 96*S1312*B1422+ 8*S1313*B1411+ 16*S1313*B1412+
48*S1321*B1411+ 96*S1321*B1412+ 8*S1322*B1411+ 16*S1322*B1412+
48*S1322*B1421+ 96*S1322*B1422+ 8*S1323*B1411+ 16*S1323*B1412+
48*S2211*B2311+ 96*S2211*B2312+ 8*S2212*B2311+ 16*S2212*B2312+
48*S2212*B2321+ 96*S2212*B2322+ 8*S2213*B2311+ 16*S2213*B2312+
48*S2221*B2311+ 96*S2221*B2312+ 8*S2222*B2311+ 16*S2222*B2312+
48*S2222*B2321+ 96*S2222*B2322+ 8*S2223*B2311+ 16*S2223*B2312+
48*S3311*B3411+ 96*S3311*B3412+ 8*S3312*B3411+ 16*S3312*B3412+
48*S3312*B3421+ 96*S3312*B3422+ 8*S3313*B3411+ 16*S3313*B3412+
48*S3321*B3411+ 96*S3321*B3412+ 8*S3322*B3411+ 16*S3322*B3412+
48*S3322*B3421+ 96*S3322*B3422+ 8*S3323*B3411+ 16*S3323*B3412+
48*S4211*B4311+ 96*S4211*B4312+ 8*S4212*B4311+ 16*S4212*B4312+
48*S4212*B4321+ 96*S4212*B4322+ 8*S4213*B4311+ 16*S4213*B4312+
41*S1111*S1212*B1211+ 82*S1111*S1212*B1212+ 51*S1111*S1213*B1211+
102*S1111*S1213*B1212+ 41*S1111*S1213*B1221+ 82*S1111*S1213*B1222+
41*S1112*S1211*B1211+ 82*S1112*S1211*B1212+ 46*S1112*S1213*B1221+
92*S1112*S1213*B1222+ 51*S1113*S1211*B1211+ 102*S1113*S1211*B1212+
41*S1113*S1211*B1221+ 82*S1113*S1211*B1222+ 46*S1113*S1212*B1221+
92*S1113*S1212*B1222+ 41*S1121*S1212*B1211+ 82*S1121*S1212*B1212+
51*S1121*S1213*B1211+ 102*S1121*S1213*B1212+ 41*S1121*S1213*B1221+
82*S1121*S1213*B1222+ 41*S1122*S1211*B1211+ 82*S1122*S1211*B1212+
46*S1122*S1213*B1221+ 92*S1122*S1213*B1222+ 51*S1123*S1211*B1211+
102*S1123*S1211*B1212+ 41*S1123*S1211*B1221+ 82*S1123*S1211*B1222+
46*S1123*S1212*B1221+ 92*S1123*S1212*B1222+ 41*S1211*S1312*B1311+
```

82*S1211*S1312*B1312+ 51*S1211*S1313*B1311+ 102*S1211*S1313*B1312+
41*S1211*S1313*B1321+ 82*S1211*S1313*B1322+ 41*S1212*S1311*B1311+
82*S1212*S1311*B1312+ 46*S1212*S1313*B1321+ 92*S1212*S1313*B1322+
51*S1213*S1311*B1311+ 102*S1213*S1311*B1312+ 41*S1213*S1311*B1321+
82*S1213*S1311*B1322+ 46*S1213*S1312*B1321+ 92*S1213*S1312*B1322+
41*S1211*S1322*B1311+ 82*S1211*S1322*B1312+ 51*S1211*S1323*B1311+
102*S1211*S1323*B1312+ 41*S1211*S1323*B1321+ 82*S1211*S1323*B1322+
41*S1212*S1321*B1311+ 82*S1212*S1321*B1312+ 46*S1212*S1323*B1321+
92*S1212*S1323*B1322+ 51*S1213*S1321*B1311+ 102*S1213*S1321*B1312+
41*S1213*S1321*B1321+ 82*S1213*S1321*B1322+ 46*S1213*S1322*B1321+
92*S1213*S1322*B1322+ 41*S2111*S2212*B2211+ 82*S2111*S2212*B2212+
51*S2111*S2213*B2211+ 102*S2111*S2213*B2212+ 41*S2111*S2213*B2221+
82*S2111*S2213*B2222+ 41*S2112*S2211*B2211+ 82*S2112*S2211*B2212+
46*S2112*S2213*B2221+ 92*S2112*S2213*B2222+ 51*S2113*S2211*B2211+
102*S2113*S2211*B2212+ 41*S2113*S2211*B2221+ 82*S2113*S2211*B2222+
46*S2113*S2212*B2221+ 92*S2113*S2212*B2222+ 41*S2111*S2222*B2211+
82*S2111*S2222*B2212+ 51*S2111*S2223*B2211+ 102*S2111*S2223*B2212+
41*S2111*S2223*B2221+ 82*S2111*S2223*B2222+ 41*S2112*S2221*B2211+
82*S2112*S2221*B2212+ 46*S2112*S2223*B2221+ 92*S2112*S2223*B2222+
51*S2113*S2221*B2211+ 102*S2113*S2221*B2212+ 41*S2113*S2221*B2221+
82*S2113*S2221*B2222+ 46*S2113*S2222*B2221+ 92*S2113*S2222*B2222+
41*S2121*S2212*B2211+ 82*S2121*S2212*B2212+ 51*S2121*S2213*B2211+
102*S2121*S2213*B2212+ 41*S2121*S2213*B2221+ 82*S2121*S2213*B2222+
41*S2122*S2211*B2211+ 82*S2122*S2211*B2212+ 46*S2122*S2213*B2221+
92*S2122*S2213*B2222+ 51*S2123*S2211*B2211+ 102*S2123*S2211*B2212+
41*S2123*S2211*B2221+ 82*S2123*S2211*B2222+ 46*S2123*S2212*B2221+
92*S2123*S2212*B2222+ 41*S2121*S2222*B2211+ 82*S2121*S2222*B2212+
51*S2121*S2223*B2211+ 102*S2121*S2223*B2212+ 41*S2121*S2223*B2221+
82*S2121*S2223*B2222+ 41*S2122*S2221*B2211+ 82*S2122*S2221*B2212+
46*S2122*S2223*B2221+ 92*S2122*S2223*B2222+ 51*S2123*S2221*B2211+
102*S2123*S2221*B2212+ 41*S2123*S2221*B2221+ 82*S2123*S2221*B2222+
46*S2123*S2222*B2221+ 92*S2123*S2222*B2222+ 41*S3111*S3212*B3211+
82*S3111*S3212*B3212+ 51*S3111*S3213*B3211+ 102*S3111*S3213*B3212+
41*S3111*S3213*B3221+ 82*S3111*S3213*B3222+ 41*S3112*S3211*B3211+
82*S3112*S3211*B3212+ 46*S3112*S3213*B3221+ 92*S3112*S3213*B3222+
51*S3113*S3211*B3211+ 102*S3113*S3211*B3212+ 41*S3113*S3211*B3221+
82*S3113*S3211*B3222+ 46*S3113*S3212*B3221+ 92*S3113*S3212*B3222+
41*S3121*S3212*B3211+ 82*S3121*S3212*B3212+ 51*S3121*S3213*B3211+
102*S3121*S3213*B3212+ 41*S3121*S3213*B3221+ 82*S3121*S3213*B3222+
41*S3122*S3211*B3211+ 82*S3122*S3211*B3212+ 46*S3122*S3213*B3221+
92*S3122*S3213*B3222+ 51*S3123*S3211*B3211+ 102*S3123*S3211*B3212+
41*S3123*S3211*B3221+ 82*S3123*S3211*B3222+ 46*S3123*S3212*B3221+
92*S3123*S3212*B3222+ 41*S3211*S3312*B3311+ 82*S3211*S3312*B3312+
51*S3211*S3313*B3311+ 102*S3211*S3313*B3312+ 41*S3211*S3313*B3321+
82*S3211*S3313*B3322+ 41*S3212*S3311*B3311+ 82*S3212*S3311*B3312+
46*S3212*S3313*B3321+ 92*S3212*S3313*B3322+ 51*S3213*S3311*B3311+
102*S3213*S3311*B3312+ 41*S3213*S3311*B3321+ 82*S3213*S3311*B3322+
46*S3213*S3312*B3321+ 92*S3213*S3312*B3322+ 41*S3211*S3322*B3311+
82*S3211*S3322*B3312+ 51*S3211*S3323*B3311+ 102*S3211*S3323*B3312+
41*S3211*S3323*B3321+ 82*S3211*S3323*B3322+ 41*S3212*S3321*B3311+
82*S3212*S3321*B3312+ 46*S3212*S3323*B3321+ 92*S3212*S3323*B3322+
51*S3213*S3321*B3311+ 102*S3213*S3321*B3312+ 41*S3213*S3321*B3321+
82*S3213*S3321*B3322+ 46*S3213*S3322*B3321+ 92*S3213*S3322*B3322+
41*S4111*S4212*B4211+ 82*S4111*S4212*B4212+ 51*S4111*S4213*B4211+
102*S4111*S4213*B4212+ 41*S4111*S4213*B4221+ 82*S4111*S4213*B4222+
41*S4112*S4211*B4211+ 82*S4112*S4211*B4212+ 46*S4112*S4213*B4221+
92*S4112*S4213*B4222+ 51*S4113*S4211*B4211+ 102*S4113*S4211*B4212+
41*S4113*S4211*B4221+ 82*S4113*S4211*B4222+ 46*S4113*S4212*B4221+
92*S4113*S4212*B4222+ 41*S4121*S4212*B4211+ 82*S4121*S4212*B4212+
51*S4121*S4213*B4211+ 102*S4121*S4213*B4212+ 41*S4121*S4213*B4221+
82*S4121*S4213*B4222+ 41*S4122*S4211*B4211+ 82*S4122*S4211*B4212+
46*S4122*S4213*B4221+ 92*S4122*S4213*B4222+ 51*S4123*S4211*B4211+
102*S4123*S4211*B4212+ 41*S4123*S4211*B4221+ 82*S4123*S4211*B4222+
46*S4123*S4212*B4221+ 92*S4123*S4212*B4222 <= 10000;
!AGV 2;
16*S1112*B1111+ 32*S1112*B1112+ 32*S1112*B1121+ 64*S1112*B1122+
16*S1122*B1111+ 32*S1122*B1112+ 32*S1122*B1121+ 64*S1122*B1122+
16*S2112*B2111+ 32*S2112*B2112+ 32*S2112*B2121+ 64*S2112*B2122+
16*S2122*B2111+ 32*S2122*B2112+ 32*S2122*B2121+ 64*S2122*B2122+
16*S3112*B3111+ 32*S3112*B3112+ 32*S3112*B3121+ 64*S3112*B3122+

16*S3122*B3111+ 32*S3122*B3112+ 32*S3122*B3121+ 64*S3122*B3122+
16*S4112*B4111+ 32*S4112*B4112+ 32*S4112*B4121+ 64*S4112*B4122+
16*S4122*B4111+ 32*S4122*B4112+ 32*S4122*B4121+ 64*S4122*B4122+
32*S1312*B1411+ 64*S1312*B1412+ 16*S1312*B1421+ 32*S1312*B1422+
32*S1322*B1411+ 64*S1322*B1412+ 16*S1322*B1421+ 32*S1322*B1422+
32*S2212*B2311+ 64*S2212*B2312+ 16*S2212*B2321+ 32*S2212*B2322+
32*S2222*B2311+ 64*S2222*B2312+ 16*S2222*B2321+ 32*S2222*B2322+
32*S3312*B3411+ 64*S3312*B3412+ 16*S3312*B3421+ 32*S3312*B3422+
32*S3322*B3411+ 64*S3322*B3412+ 16*S3322*B3421+ 32*S3322*B3422+
32*S4212*B4311+ 64*S4212*B4312+ 16*S4212*B4321+ 32*S4212*B4322+
16*S1111*S1212*B1211+ 32*S1111*S1212*B1212+ 33*S1111*S1213*B1221+
66*S1111*S1213*B1222+ 16*S1112*S1211*B1211+ 32*S1112*S1211*B1212+
32*S1112*S1213*B1211+ 64*S1112*S1213*B1212+ 16*S1112*S1213*B1221+
32*S1112*S1213*B1222+ 33*S1113*S1211*B1221+ 66*S1113*S1211*B1222+
32*S1113*S1212*B1211+ 64*S1113*S1212*B1212+ 16*S1113*S1212*B1221+
32*S1113*S1212*B1222+ 16*S1121*S1212*B1211+ 32*S1121*S1212*B1212+
33*S1121*S1213*B1221+ 66*S1121*S1213*B1222+ 16*S1122*S1211*B1211+
32*S1122*S1211*B1212+ 32*S1122*S1213*B1211+ 64*S1122*S1213*B1212+
16*S1122*S1213*B1221+ 32*S1122*S1213*B1222+ 33*S1123*S1211*B1221+
66*S1123*S1211*B1222+ 32*S1123*S1212*B1211+ 64*S1123*S1212*B1212+
16*S1123*S1212*B1221+ 32*S1123*S1212*B1222+ 16*S1211*S1312*B1311+
32*S1211*S1312*B1312+ 33*S1211*S1313*B1321+ 66*S1211*S1313*B1322+
16*S1212*S1311*B1311+ 32*S1212*S1311*B1312+ 32*S1212*S1313*B1311+
64*S1212*S1313*B1312+ 16*S1212*S1313*B1321+ 32*S1212*S1313*B1322+
33*S1213*S1311*B1321+ 66*S1213*S1311*B1322+ 32*S1213*S1312*B1311+
64*S1213*S1312*B1312+ 16*S1213*S1312*B1321+ 32*S1213*S1312*B1322+
16*S1211*S1322*B1311+ 32*S1211*S1322*B1312+ 33*S1211*S1323*B1321+
66*S1211*S1323*B1322+ 16*S1212*S1321*B1311+ 32*S1212*S1321*B1312+
32*S1212*S1323*B1311+ 64*S1212*S1323*B1312+ 16*S1212*S1323*B1321+
32*S1212*S1323*B1322+ 33*S1213*S1321*B1321+ 66*S1213*S1321*B1322+
32*S1213*S1322*B1311+ 64*S1213*S1322*B1312+ 16*S1213*S1322*B1321+
32*S1213*S1322*B1322+ 16*S2111*S2212*B2211+ 32*S2111*S2212*B2212+
33*S2111*S2213*B2221+ 66*S2111*S2213*B2222+ 16*S2112*S2211*B2211+
32*S2112*S2211*B2212+ 32*S2112*S2213*B2211+ 64*S2112*S2213*B2212+
16*S2112*S2213*B2221+ 32*S2112*S2213*B2222+ 33*S2113*S2211*B2221+
66*S2113*S2211*B2222+ 32*S2113*S2212*B2211+ 64*S2113*S2212*B2212+
16*S2113*S2212*B2221+ 32*S2113*S2212*B2222+ 16*S2111*S2222*B2211+
32*S2111*S2222*B2212+ 33*S2111*S2223*B2221+ 66*S2111*S2223*B2222+
16*S2112*S2221*B2211+ 32*S2112*S2221*B2212+ 32*S2112*S2223*B2211+
64*S2112*S2223*B2212+ 16*S2112*S2223*B2221+ 32*S2112*S2223*B2222+
33*S2113*S2221*B2221+ 66*S2113*S2221*B2222+ 32*S2113*S2222*B2211+
64*S2113*S2222*B2212+ 16*S2113*S2222*B2221+ 32*S2113*S2222*B2222+
16*S2121*S2212*B2211+ 32*S2121*S2212*B2212+ 33*S2121*S2213*B2221+
66*S2121*S2213*B2222+ 16*S2122*S2211*B2211+ 32*S2122*S2211*B2212+
32*S2122*S2213*B2211+ 64*S2122*S2213*B2212+ 16*S2122*S2213*B2221+
32*S2122*S2213*B2222+ 33*S2123*S2211*B2221+ 66*S2123*S2211*B2222+
32*S2123*S2212*B2211+ 64*S2123*S2212*B2212+ 16*S2123*S2212*B2221+
32*S2123*S2212*B2222+ 16*S2121*S2222*B2211+ 32*S2121*S2222*B2212+
33*S2121*S2223*B2221+ 66*S2121*S2223*B2222+ 16*S2122*S2221*B2211+
32*S2122*S2221*B2212+ 32*S2122*S2223*B2211+ 64*S2122*S2223*B2212+
16*S2122*S2223*B2221+ 32*S2122*S2223*B2222+ 33*S2123*S2221*B2221+
66*S2123*S2221*B2222+ 32*S2123*S2222*B2211+ 64*S2123*S2222*B2212+
16*S2123*S2222*B2221+ 32*S2123*S2222*B2222+ 16*S3111*S3212*B3211+
32*S3111*S3212*B3212+ 33*S3111*S3213*B3221+ 66*S3111*S3213*B3222+
16*S3112*S3211*B3211+ 32*S3112*S3211*B3212+ 32*S3112*S3213*B3211+
64*S3112*S3213*B3212+ 16*S3112*S3213*B3221+ 32*S3112*S3213*B3222+
33*S3113*S3211*B3221+ 66*S3113*S3211*B3222+ 32*S3113*S3212*B3211+
64*S3113*S3212*B3212+ 16*S3113*S3212*B3221+ 32*S3113*S3212*B3222+
16*S3121*S3212*B3211+ 32*S3121*S3212*B3212+ 33*S3121*S3213*B3221+
66*S3121*S3213*B3222+ 16*S3122*S3211*B3211+ 32*S3122*S3211*B3212+
32*S3122*S3213*B3211+ 64*S3122*S3213*B3212+ 16*S3122*S3213*B3221+
32*S3122*S3213*B3222+ 33*S3123*S3211*B3221+ 66*S3123*S3211*B3222+
32*S3123*S3212*B3211+ 64*S3123*S3212*B3212+ 16*S3123*S3212*B3221+
32*S3123*S3212*B3222+ 16*S3211*S3312*B3311+ 32*S3211*S3312*B3312+
33*S3211*S3313*B3321+ 66*S3211*S3313*B3322+ 16*S3212*S3311*B3311+
32*S3212*S3311*B3312+ 32*S3212*S3313*B3311+ 64*S3212*S3313*B3312+
16*S3212*S3313*B3321+ 32*S3212*S3313*B3322+ 33*S3213*S3311*B3321+
66*S3213*S3311*B3322+ 32*S3213*S3312*B3311+ 64*S3213*S3312*B3312+
16*S3213*S3312*B3321+ 32*S3213*S3312*B3322+ 16*S3211*S3322*B3311+
32*S3211*S3322*B3312+ 33*S3211*S3323*B3321+ 66*S3211*S3323*B3322+

16*S3212*S3321*B3311+ 32*S3212*S3321*B3312+ 32*S3212*S3323*B3311+
64*S3212*S3323*B3312+ 16*S3212*S3323*B3321+ 32*S3212*S3323*B3322+
33*S3213*S3321*B3321+ 66*S3213*S3321*B3322+ 32*S3213*S3322*B3311+
64*S3213*S3322*B3312+ 16*S3213*S3322*B3321+ 32*S3213*S3322*B3322+
16*S4111*S4212*B4211+ 32*S4111*S4212*B4212+ 33*S4111*S4213*B4221+
66*S4111*S4213*B4222+ 16*S4112*S4211*B4211+ 32*S4112*S4211*B4212+
32*S4112*S4213*B4211+ 64*S4112*S4213*B4212+ 16*S4112*S4213*B4221+
32*S4112*S4213*B4222+ 33*S4113*S4211*B4221+ 66*S4113*S4211*B4222+
32*S4113*S4212*B4211+ 64*S4113*S4212*B4212+ 16*S4113*S4212*B4221+
32*S4113*S4212*B4222+ 16*S4121*S4212*B4211+ 32*S4121*S4212*B4212+
33*S4121*S4213*B4221+ 66*S4121*S4213*B4222+ 16*S4122*S4211*B4211+
32*S4122*S4211*B4212+ 32*S4122*S4213*B4211+ 64*S4122*S4213*B4212+
16*S4122*S4213*B4221+ 32*S4122*S4213*B4222+ 33*S4123*S4211*B4221+
66*S4123*S4211*B4222+ 32*S4123*S4212*B4211+ 64*S4123*S4212*B4212+
16*S4123*S4212*B4221+ 32*S4123*S4212*B4222 <= 10000;
!AGV 3;
33*S1112*B1121+ 66*S1112*B1122+ 16*S1113*B1111+ 32*S1113*B1112+
33*S1122*B1121+ 66*S1122*B1122+ 16*S1123*B1111+ 32*S1123*B1112+
33*S2112*B2121+ 66*S2112*B2122+ 16*S2113*B2111+ 32*S2113*B2112+
33*S2122*B2121+ 66*S2122*B2122+ 16*S2123*B2111+ 32*S2123*B2112+
33*S3112*B3121+ 66*S3112*B3122+ 16*S3113*B3111+ 32*S3113*B3112+
33*S3122*B3121+ 66*S3122*B3122+ 16*S3123*B3111+ 32*S3123*B3112+
33*S4112*B4121+ 66*S4112*B4122+ 16*S4113*B4111+ 32*S4113*B4112+
33*S4122*B4121+ 66*S4122*B4122+ 16*S4123*B4111+ 32*S4123*B4112+
33*S1312*B1411+ 66*S1312*B1412+ 16*S1313*B1411+ 32*S1313*B1412+
33*S1322*B1411+ 66*S1322*B1412+ 16*S1323*B1411+ 32*S1323*B1412+
33*S2212*B2311+ 66*S2212*B2312+ 16*S2213*B2311+ 32*S2213*B2312+
33*S2222*B2311+ 66*S2222*B2312+ 16*S2223*B2311+ 32*S2223*B2312+
33*S3312*B3411+ 66*S3312*B3412+ 16*S3313*B3411+ 32*S3313*B3412+
33*S3322*B3411+ 66*S3322*B3412+ 16*S3323*B3411+ 32*S3323*B3412+
33*S4212*B4311+ 66*S4212*B4312+ 16*S4213*B4311+ 32*S4213*B4312+
16*S1111*S1213*B1211+ 32*S1111*S1213*B1212+ 32*S1111*S1213*B1221+
64*S1111*S1213*B1222+ 32*S1112*S1213*B1211+ 64*S1112*S1213*B1212+
16*S1112*S1213*B1221+ 32*S1112*S1213*B1222+ 16*S1113*S1211*B1211+
32*S1113*S1211*B1212+ 32*S1113*S1211*B1221+ 64*S1113*S1211*B1222+
32*S1113*S1212*B1211+ 64*S1113*S1212*B1212+ 16*S1113*S1212*B1221+
32*S1113*S1212*B1222+ 16*S1121*S1213*B1211+ 32*S1121*S1213*B1212+
32*S1121*S1213*B1221+ 64*S1121*S1213*B1222+ 32*S1122*S1213*B1211+
64*S1122*S1213*B1212+ 16*S1122*S1213*B1221+ 32*S1122*S1213*B1222+
16*S1123*S1211*B1211+ 32*S1123*S1211*B1212+ 32*S1123*S1211*B1221+
64*S1123*S1211*B1222+ 32*S1123*S1212*B1211+ 64*S1123*S1212*B1212+
16*S1123*S1212*B1221+ 32*S1123*S1212*B1222+ 16*S1211*S1313*B1311+
32*S1211*S1313*B1312+ 32*S1211*S1313*B1321+ 64*S1211*S1313*B1322+
32*S1212*S1313*B1311+ 64*S1212*S1313*B1312+ 16*S1212*S1313*B1321+
32*S1212*S1313*B1322+ 16*S1213*S1311*B1311+ 32*S1213*S1311*B1312+
32*S1213*S1311*B1321+ 64*S1213*S1311*B1322+ 32*S1213*S1312*B1311+
64*S1213*S1312*B1312+ 16*S1213*S1312*B1321+ 32*S1213*S1312*B1322+
16*S1211*S1323*B1311+ 32*S1211*S1323*B1312+ 32*S1211*S1323*B1321+
64*S1211*S1323*B1322+ 32*S1212*S1323*B1311+ 64*S1212*S1323*B1312+
16*S1212*S1323*B1321+ 32*S1212*S1323*B1322+ 16*S1213*S1321*B1311+
32*S1213*S1321*B1312+ 32*S1213*S1321*B1321+ 64*S1213*S1321*B1322+
32*S1213*S1322*B1311+ 64*S1213*S1322*B1312+ 16*S1213*S1322*B1321+
32*S1213*S1322*B1322+ 16*S2111*S2213*B2211+ 32*S2111*S2213*B2212+
32*S2111*S2213*B2221+ 64*S2111*S2213*B2222+ 32*S2112*S2213*B2211+
64*S2112*S2213*B2212+ 16*S2112*S2213*B2221+ 32*S2112*S2213*B2222+
16*S2113*S2211*B2211+ 32*S2113*S2211*B2212+ 32*S2113*S2211*B2221+
64*S2113*S2211*B2222+ 32*S2113*S2212*B2211+ 64*S2113*S2212*B2212+
16*S2113*S2212*B2221+ 32*S2113*S2212*B2222+ 16*S2111*S2223*B2211+
32*S2111*S2223*B2212+ 32*S2111*S2223*B2221+ 64*S2111*S2223*B2222+
32*S2112*S2223*B2211+ 64*S2112*S2223*B2212+ 16*S2112*S2223*B2221+
32*S2112*S2223*B2222+ 16*S2113*S2221*B2211+ 32*S2113*S2221*B2212+
32*S2113*S2221*B2221+ 64*S2113*S2221*B2222+ 32*S2113*S2222*B2211+
64*S2113*S2222*B2212+ 16*S2113*S2222*B2221+ 32*S2113*S2222*B2222+
16*S2121*S2213*B2211+ 32*S2121*S2213*B2212+ 32*S2121*S2213*B2221+
64*S2121*S2213*B2222+ 32*S2122*S2213*B2211+ 64*S2122*S2213*B2212+
16*S2122*S2213*B2221+ 32*S2122*S2213*B2222+ 16*S2123*S2211*B2211+
32*S2123*S2211*B2212+ 32*S2123*S2211*B2221+ 64*S2123*S2211*B2222+
32*S2123*S2212*B2211+ 64*S2123*S2212*B2212+ 16*S2123*S2212*B2221+
32*S2123*S2212*B2222+ 16*S2121*S2223*B2211+ 32*S2121*S2223*B2212+
32*S2121*S2223*B2221+ 64*S2121*S2223*B2222+ 32*S2122*S2223*B2211+

```
64*S2122*S2223*B2212+ 16*S2122*S2223*B2221+ 32*S2122*S2223*B2222+
16*S2123*S2221*B2211+ 32*S2123*S2221*B2212+ 32*S2123*S2221*B2221+
64*S2123*S2221*B2222+ 32*S2123*S2222*B2211+ 64*S2123*S2222*B2212+
16*S2123*S2222*B2221+ 32*S2123*S2222*B2222+ 16*S3111*S3213*B3211+
32*S3111*S3213*B3212+ 32*S3111*S3213*B3221+ 64*S3111*S3213*B3222+
32*S3112*S3213*B3211+ 64*S3112*S3213*B3212+ 16*S3112*S3213*B3221+
32*S3112*S3213*B3222+ 16*S3113*S3211*B3211+ 32*S3113*S3211*B3212+
32*S3113*S3211*B3221+ 64*S3113*S3211*B3222+ 32*S3113*S3212*B3211+
64*S3113*S3212*B3212+ 16*S3113*S3212*B3221+ 32*S3113*S3212*B3222+
16*S3121*S3213*B3211+ 32*S3121*S3213*B3212+ 32*S3121*S3213*B3221+
64*S3121*S3213*B3222+ 32*S3122*S3213*B3211+ 64*S3122*S3213*B3212+
16*S3122*S3213*B3221+ 32*S3122*S3213*B3222+ 16*S3123*S3211*B3211+
32*S3123*S3211*B3212+ 32*S3123*S3211*B3221+ 64*S3123*S3211*B3222+
32*S3123*S3212*B3211+ 64*S3123*S3212*B3212+ 16*S3123*S3212*B3221+
32*S3123*S3212*B3222+ 16*S3211*S3313*B3311+ 32*S3211*S3313*B3312+
32*S3211*S3313*B3321+ 64*S3211*S3313*B3322+ 32*S3212*S3313*B3311+
64*S3212*S3313*B3312+ 16*S3212*S3313*B3321+ 32*S3212*S3313*B3322+
16*S3213*S3311*B3311+ 32*S3213*S3311*B3312+ 32*S3213*S3311*B3321+
64*S3213*S3311*B3322+ 32*S3213*S3312*B3311+ 64*S3213*S3312*B3312+
16*S3213*S3312*B3321+ 32*S3213*S3312*B3322+ 16*S3211*S3323*B3311+
32*S3211*S3323*B3312+ 32*S3211*S3323*B3321+ 64*S3211*S3323*B3322+
32*S3212*S3323*B3311+ 64*S3212*S3323*B3312+ 16*S3212*S3323*B3321+
32*S3212*S3323*B3322+ 16*S3213*S3321*B3311+ 32*S3213*S3321*B3312+
32*S3213*S3321*B3321+ 64*S3213*S3321*B3322+ 32*S3213*S3322*B3311+
64*S3213*S3322*B3312+ 16*S3213*S3322*B3321+ 32*S3213*S3322*B3322+
16*S4111*S4213*B4211+ 32*S4111*S4213*B4212+ 32*S4111*S4213*B4221+
64*S4111*S4213*B4222+ 32*S4112*S4213*B4211+ 64*S4112*S4213*B4212+
16*S4112*S4213*B4221+ 32*S4112*S4213*B4222+ 16*S4113*S4211*B4211+
32*S4113*S4211*B4212+ 32*S4113*S4211*B4221+ 64*S4113*S4211*B4222+
32*S4113*S4212*B4211+ 64*S4113*S4212*B4212+ 16*S4113*S4212*B4221+
32*S4113*S4212*B4222+ 16*S4121*S4213*B4211+ 32*S4121*S4213*B4212+
32*S4121*S4213*B4221+ 64*S4121*S4213*B4222+ 32*S4122*S4213*B4211+
64*S4122*S4213*B4212+ 16*S4122*S4213*B4221+ 32*S4122*S4213*B4222+
16*S4123*S4211*B4211+ 32*S4123*S4211*B4212+ 32*S4123*S4211*B4221+
64*S4123*S4211*B4222+ 32*S4123*S4212*B4211+ 64*S4123*S4212*B4212+
16*S4123*S4212*B4221+ 32*S4123*S4212*B4222 <= 10000;
! End of OPERATIONAL CONSTRAINTS;

!Declare the S Variables as Binary Integer Variables;
@BIN(S1111);  @BIN(S1112);  @BIN(S1113);  @BIN(S1121);  @BIN(S1122);  @BIN(S1123);
@BIN(S1211);  @BIN(S1212);  @BIN(S1213);  @BIN(S1311);  @BIN(S1312);  @BIN(S1313);
@BIN(S1321);  @BIN(S1322);  @BIN(S1323);  @BIN(S2111);  @BIN(S2112);  @BIN(S2113);
@BIN(S2121);  @BIN(S2122);  @BIN(S2123);  @BIN(S2211);  @BIN(S2212);  @BIN(S2213);
@BIN(S2221);  @BIN(S2222);  @BIN(S2223);  @BIN(S3111);  @BIN(S3112);  @BIN(S3113);
@BIN(S3121);  @BIN(S3122);  @BIN(S3123);  @BIN(S3211);  @BIN(S3212);  @BIN(S3213);
@BIN(S3311);  @BIN(S3312);  @BIN(S3313);  @BIN(S3321);  @BIN(S3322);  @BIN(S3323);
@BIN(S4111);  @BIN(S4112);  @BIN(S4113);  @BIN(S4121);  @BIN(S4122);  @BIN(S4123);
@BIN(S4211);  @BIN(S4212);  @BIN(S4213);

!Declare the B Variables as Binary Integer Variables;
@BIN(B1111);  @BIN(B1112);  @BIN(B1121);  @BIN(B1122);  @BIN(B1211);  @BIN(B1212);
@BIN(B1221);  @BIN(B1222);  @BIN(B1311);  @BIN(B1312);  @BIN(B1321);  @BIN(B1322);
@BIN(B1411);  @BIN(B1412);  @BIN(B1421);  @BIN(B1422);  @BIN(B2111);  @BIN(B2112);
@BIN(B2121);  @BIN(B2122);  @BIN(B2211);  @BIN(B2212);  @BIN(B2221);  @BIN(B2222);
@BIN(B2311);  @BIN(B2312);  @BIN(B2321);  @BIN(B2322);  @BIN(B3111);  @BIN(B3112);
@BIN(B3121);  @BIN(B3122);  @BIN(B3211);  @BIN(B3212);  @BIN(B3221);  @BIN(B3222);
@BIN(B3311);  @BIN(B3312);  @BIN(B3321);  @BIN(B3322);  @BIN(B3411);  @BIN(B3412);
@BIN(B3421);  @BIN(B3422);  @BIN(B4111);  @BIN(B4112);  @BIN(B4121);  @BIN(B4122);
@BIN(B4211);  @BIN(B4212);  @BIN(B4221);  @BIN(B4222);  @BIN(B4311);  @BIN(B4312);
@BIN(B4321);  @BIN(B4322);

!Declare the X Variables as Binary Integer Variables;
@BIN(X111);  @BIN(X112);  @BIN(X113);  @BIN(X121);  @BIN(X122);  @BIN(X123);
@BIN(X211);  @BIN(X212);  @BIN(X213);  @BIN(X221);  @BIN(X222);  @BIN(X223);
@BIN(X311);  @BIN(X312);  @BIN(X313);
@BIN(X411);  @BIN(X412);  @BIN(X413);  @BIN(X421);  @BIN(X422);  @BIN(X423);
@BIN(X511);  @BIN(X512);  @BIN(X513);

END
!End of MODEL;
```

APPENDIX A.3

Final Model of the First Problem Instance

Note: This mathematical model is the result of applying the first and second transformation techniques to the original model presented in Appendix A.1. The first transformation technique transforms all the general integer variables into binary variables. The second transformation technique transforms all the non-linear terms into their linear counterparts. The second technique used the approach proposed by Glover and Woolsey (1972). Notice that in this model, all the cross product terms that appear in the model presented in Appendix A.2, are replaced with new real variables. The model is a mixed integer linear programming model with all its real variables bounded between 0 and 1.

! OBJECTIVE FUNCTION;

MODEL:

MIN =

!Part 1 - Operation 1;
30*S1B1 + 60*S1B2 + 46*S2B1 + 92*S2B2 + 58*S2B3 + 116*S2B4 + 21*S3B1 + 42*S3B2 + 30*S4B1 + 60*S4B2 + 46*S5B1 + 92*S5B2 + 58*S5B3 + 116*S5B4 + 21*S6B1 + 42*S6B2 +
!Part 2 - Operation 1;
30*S19B17 + 60*S19B18 + 46*S20B17 + 92*S20B18 + 58*S20B19 + 116*S20B20 + 21*S21B17 + 42*S21B18 + 30*S22B17 + 60*S22B18 + 46*S23B17 + 92*S23B18 + 58*S23B19 + 116*S23B20 + 21*S24B17 + 42*S24B18 +
!Part 3 - Operation 1;
30*S37B33 + 60*S37B34 + 46*S38B33 + 92*S38B34 + 58*S38B35 + 116*S38B36 + 21*S39B33 + 42*S39B34 + 30*S40B33 + 60*S40B34 + 46*S41B33 + 92*S41B34 + 58*S41B35 + 116*S41B36 + 21*S42B33 + 42*S42B34 +
!Part 4 - Operation 1;
30*S55B49 + 60*S55B50 + 46*S56B49 + 92*S56B50 + 58*S56B51 + 116*S56B52 + 21*S57B49 + 42*S57B50 + 30*S58B49 + 60*S58B50 + 46*S59B49 + 92*S59B50 + 58*S59B51 + 116*S59B52 + 21*S60B49 + 42*S60B50 +
!Part 1 - Operation 4;
48*S13B13 + 96*S13B14 + 55*S14B13 + 110*S14B14 + 58*S14B15 + 116*S14B16 + 18*S15B13 + 36*S15B14 + 48*S16B13 + 96*S16B14 + 55*S17B13 + 110*S17B14 + 58*S17B15 + 116*S17B16 + 18*S18B13 + 36*S18B14 +
!Part 2 - Operation 3;
48*S25B25 + 96*S25B26 + 55*S26B25 + 110*S26B26 + 58*S26B27 + 116*S26B28 + 18*S27B25 + 36*S27B26 + 48*S28B25 + 96*S28B26 + 55*S29B25 + 110*S29B26 + 58*S29B27 + 116*S29B28 + 18*S30B25 + 36*S30B26 +
!Part 3 - Operation 4;
48*S49B45 + 96*S49B46 + 55*S50B45 + 110*S50B46 + 58*S50B47 + 116*S50B48 + 18*S51B45 + 36*S51B46 + 48*S52B45 + 96*S52B46 + 55*S53B45 + 110*S53B46 + 58*S53B47 + 116*S53B48 + 18*S54B45 + 36*S54B46 +
!Part 4 - Operation 3;
48*S61B57 + 96*S61B58 + 55*S62B57 + 110*S62B58 + 58*S62B59 + 116*S62B60 + 18*S63B57 + 36*S63B58 +
!Part 1 - Operation 2;
39*S1S8B5 + 78*S1S8B6 + 64*S1S9B5 + 128*S1S9B6 + 76*S1S9B7 + 152*S1S9B8 + 33*S2S7B5 + 66*S2S7B6 + 52*S2S9B5 + 104*S2S9B6 + 69*S2S9B7 + 138*S2S9B8 + 43*S3S7B5 + 86*S3S7B6 + 70*S3S7B7 + 140*S3S7B8 + 52*S3S8B5 + 104*S3S8B6 + 54*S3S8B7 + 108*S3S8B8 + 39*S4S8B5 + 78*S4S8B6 + 64*S4S9B5 + 128*S4S9B6 + 76*S4S9B7 + 152*S4S9B8 + 33*S5S7B5 + 66*S5S7B6 + 52*S5S9B5 + 104*S5S9B6 + 69*S5S9B7 + 138*S5S9B8 + 43*S6S7B5 + 86*S6S7B6 + 70*S6S7B7 + 140*S6S7B8 + 52*S6S8B5 + 104*S6S8B6 + 54*S6S8B7 + 108*S6S8B8 +
!Part 1 - Operation 3;
39*S7S14B9 + 78*S7S14B10 + 64*S7S15B9 + 128*S7S15B10 + 76*S7S15B11 + 152*S7S15B12 + 33*S8S13B9 + 66*S8S13B10 + 52*S8S15B9 + 104*S8S15B10 + 69*S8S15B11 + 138*S8S15B12 + 43*S9S13B9 + 86*S9S13B10 + 70*S9S13B11 + 40*S9S13B12 + 52*S9S14B9 + 104*S9S14B10 + 54*S9S14B11 + 108*S9S14B12 + 39*S7S17B9 + 78*S7S17B10 + 64*S7S18B9 + 128*S7S18B10 + 76*S7S18B11 + 152*S7S18B12 + 33*S8S16B9 + 66*S8S16B10 + 52*S8S18B9 + 104*S8S18B10 + 69*S8S18B11 + 138*S8S18B12 + 43*S9S16B9 + 86*S9S16B10 + 70*S9S16B11 + 140*S9S16B12 + 52*S9S17B9 + 104*S9S17B10 + 54*S9S17B11 + 108*S9S17B12 +
!Part 2 - Operation 2;
39*S19S26B21 + 78*S19S26B22 + 64*S19S27B21 + 128*S19S27B22 + 76*S19S27B23 + 152*S19S27B24 + 33*S20S25B21 + 66*S20S25B22 + 52*S20S27B21 + 104*S20S27B22 + 69*S20S27B23 + 138*S20S27B24 + 43*S21S25B21 + 86*S21S25B22 + 70*S21S25B23 + 140*S21S25B24 + 52*S21S26B21 + 104*S21S26B22 + 54*S21S26B23 + 108*S21S26B24 + 39*S19S29B21 +

78*S19S29B22 + 64*S19S30B21 + 128*S19S30B22 + 76*S19S30B23 + 152*S19S30B24 + 33*S20S28B21 + 66*S20S28B22 +
52*S20S30B21 + 104*S20S30B22 + 69*S20S30B23 + 138*S20S30B24 + 43*S21S28B21 + 86*S21S28B22 + 70*S21S28B23 +
140*S21S28B24 + 52*S21S29B21 + 104*S21S29B22 + 54*S21S29B23 + 108*S21S29B24 + 39*S22S26B21 + 78*S22S26B22 +
64*S22S27B21 + 128*S22S27B22 + 76*S22S27B23 + 152*S22S27B24 + 33*S23S25B21 + 66*S23S25B22 + 52*S23S27B21 +
104*S23S27B22 + 69*S23S27B23 + 138*S23S27B24 + 43*S24S25B21 + 86*S24S25B22 + 70*S24S25B23 + 140*S24S25B24 +
52*S24S26B21 + 104*S24S26B22 + 54*S24S26B23 + 108*S24S26B24 + 39*S22S29B21 + 78*S22S29B22 + 64*S22S30B21 +
128*S22S30B22 + 76*S22S30B23 + 152*S22S30B24 + 33*S23S28B21 + 66*S23S28B22 + 52*S23S30B21 + 104*S23S30B22 +
69*S23S30B23 + 138*S23S30B24 + 43*S24S28B21 + 86*S24S28B22 + 70*S24S28B23 + 140*S24S28B24 + 52*S24S29B21 +
104*S24S29B22 + 54*S24S29B23 + 108*S24S29B24 +
!Part 3 - Operation 2;
39*S37S44B37 + 78*S37S44B38 + 64*S37S45B37 + 128*S37S45B38 + 76*S37S45B39 + 152*S37S45B40 + 33*S38S43B37 +
66*S38S43B38 + 52*S38S45B37 + 104*S38S45B38 + 69*S38S45B39 + 138*S38S45B40 + 43*S39S43B37 + 86*S39S43B38 +
70*S39S43B39 + 140*S39S43B40 + 52*S39S44B37 + 104*S39S44B38 + 54*S39S44B39 + 108*S39S44B40 + 39*S40S44B37 +
78*S40S44B38 + 64*S40S45B37 + 128*S40S45B38 + 76*S40S45B39 + 152*S40S45B40 + 33*S41S43B37 + 66*S41S43B38 +
52*S41S45B37 + 104*S41S45B38 + 69*S41S45B39 + 138*S41S45B40 + 43*S42S43B37 + 86*S42S43B38 + 70*S42S43B39 +
140*S42S43B40 + 52*S42S44B37 + 104*S42S44B38 + 54*S42S44B39 + 108*S42S44B40 +
!Part 3 - Operation 3;
39*S43S50B41 + 78*S43S50B42 + 64*S43S51B41 + 128*S43S51B42 + 76*S43S51B43 + 152*S43S51B44 + 33*S44S49B41 +
66*S44S49B42 + 52*S44S51B41 + 104*S44S51B42 + 69*S44S51B43 + 138*S44S51B44 + 43*S45S49B41 + 86*S45S49B42 +
70*S45S49B43 + 140*S45S49B44 + 52*S45S50B41 + 104*S45S50B42 + 54*S45S50B43 + 108*S45S50B44 + 39*S43S53B41 +
78*S43S53B42 + 64*S43S54B41 + 128*S43S54B42 + 76*S43S54B43 + 152*S43S54B44 + 33*S44S52B41 + 66*S44S52B42 +
52*S44S54B41 + 104*S44S54B42 + 69*S44S54B43 + 138*S44S54B44 + 43*S45S52B41 + 86*S45S52B42 + 70*S45S52B43 +
140*S45S52B44 + 52*S45S53B41 + 104*S45S53B42 + 54*S45S53B43 + 108*S45S53B44 +
!Part 4 - Operation 2;
39*S55S62B53 + 78*S55S62B54 + 64*S55S63B53 + 128*S55S63B54 + 76*S55S63B55 + 152*S55S63B56 + 33*S56S61B53 +
66*S56S61B54 + 52*S56S63B53 + 104*S56S63B54 + 69*S56S63B55 + 138*S56S63B56 + 43*S57S61B53 + 86*S57S61B54 +
70*S57S61B55 + 140*S57S61B56 + 52*S57S62B53 + 104*S57S62B54 + 54*S57S62B55 + 108*S57S62B56 + 39*S58S62B53 +
78*S58S62B54 + 64*S58S63B53 + 128*S58S63B54 + 76*S58S63B55 + 152*S58S63B56 + 33*S59S61B53 + 66*S59S61B54 +
52*S59S63B53 + 104*S59S63B54 + 69*S59S63B55 + 138*S59S63B56 + 43*S60S61B53 + 86*S60S61B54 + 70*S60S61B55 +
140*S60S61B56 + 52*S60S62B53 + 104*S60S62B54 + 54*S60S62B55 + 108*S60S62B56;

!TRANSFORMATION CONSTRAINTS;

!These are the first constraints imposed by the transformation proposed by Glover and Woolsey (1972). Refer to Chapter 6 for detailed explanation of these constraints;

S1111+B1111-S1B1 <= 1;
S1111+B1112-S1B2 <= 1;
S1112+B1111-S2B1 <= 1;
S1112+B1112-S2B2 <= 1;
S1112+B1121-S2B3 <= 1;
S1112+B1122-S2B4 <= 1;
S1113+B1111-S3B1 <= 1;
S1113+B1112-S3B2 <= 1;
S1121+B1111-S4B1 <= 1;
S1121+B1112-S4B2 <= 1;
S1122+B1111-S5B1 <= 1;
S1122+B1112-S5B2 <= 1;
S1122+B1121-S5B3 <= 1;
S1122+B1122-S5B4 <= 1;
S1123+B1111-S6B1 <= 1;
S1123+B1112-S6B2 <= 1;
S2111+B2111-S19B17 <= 1;
S2111+B2112-S19B18 <= 1;
S2112+B2111-S20B17 <= 1;
S2112+B2112-S20B18 <= 1;
S2112+B2121-S20B19 <= 1;
S2112+B2122-S20B20 <= 1;
S2113+B2111-S21B17 <= 1;
S2113+B2112-S21B18 <= 1;
S2121+B2111-S22B17 <= 1;
S2121+B2112-S22B18 <= 1;
S2122+B2111-S23B17 <= 1;
S2122+B2112-S23B18 <= 1;
S2122+B2121-S23B19 <= 1;
S2122+B2122-S23B20 <= 1;
S2123+B2111-S24B17 <= 1;
S2123+B2112-S24B18 <= 1;
S3111+B3111-S37B33 <= 1;
S3111+B3112-S37B34 <= 1;
S3112+B3111-S38B33 <= 1;
S3112+B3112-S38B34 <= 1;
S3112+B3121-S38B35 <= 1;
S3112+B3122-S38B36 <= 1;
S3113+B3111-S39B33 <= 1;
S3113+B3112-S39B34 <= 1;
S3121+B3111-S40B33 <= 1;
S3121+B3112-S40B34 <= 1;
S3122+B3111-S41B33 <= 1;
S3122+B3112-S41B34 <= 1;
S3122+B3121-S41B35 <= 1;
S3122+B3122-S41B36 <= 1;
S3123+B3111-S42B33 <= 1;
S3123+B3112-S42B34 <= 1;
S4111+B4111-S55B49 <= 1;
S4111+B4112-S55B50 <= 1;
S4112+B4111-S56B49 <= 1;
S4112+B4112-S56B50 <= 1;
S4112+B4121-S56B51 <= 1;
S4112+B4122-S56B52 <= 1;
S4113+B4111-S57B49 <= 1;
S4113+B4112-S57B50 <= 1;
S4121+B4111-S58B49 <= 1;
S4121+B4112-S58B50 <= 1;
S4122+B4111-S59B49 <= 1;
S4122+B4112-S59B50 <= 1;
S4122+B4121-S59B51 <= 1;
S4122+B4122-S59B52 <= 1;
S4123+B4111-S60B49 <= 1;
S4123+B4112-S60B50 <= 1;
S1311+B1411-S13B13 <= 1;
S1311+B1412-S13B14 <= 1;
S1312+B1411-S14B13 <= 1;
S1312+B1412-S14B14 <= 1;
S1312+B1421-S14B15 <= 1;
S1312+B1422-S14B16 <= 1;
S1313+B1411-S15B13 <= 1;
S1313+B1412-S15B14 <= 1;
S1321+B1411-S16B13 <= 1;
S1321+B1412-S16B14 <= 1;
S1322+B1411-S17B13 <= 1;
S1322+B1412-S17B14 <= 1;
S1322+B1421-S17B15 <= 1;
S1322+B1422-S17B16 <= 1;
S1323+B1411-S18B13 <= 1;
S1323+B1412-S18B14 <= 1;
S2211+B2311-S25B25 <= 1;
S2211+B2312-S25B26 <= 1;
S2212+B2311-S26B25 <= 1;
S2212+B2312-S26B26 <= 1;
S2212+B2321-S26B27 <= 1;
S2212+B2322-S26B28 <= 1;
S2213+B2311-S27B25 <= 1;
S2213+B2312-S27B26 <= 1;
S2221+B2311-S28B25 <= 1;
S2221+B2312-S28B26 <= 1;
S2222+B2311-S29B25 <= 1;
S2222+B2312-S29B26 <= 1;
S2222+B2321-S29B27 <= 1;
S2222+B2322-S29B28 <= 1;
S2223+B2311-S30B25 <= 1;
S2223+B2312-S30B26 <= 1;
S3311+B3411-S49B45 <= 1;
S3311+B3412-S49B46 <= 1;
S3312+B3411-S50B45 <= 1;
S3312+B3412-S50B46 <= 1;
S3312+B3421-S50B47 <= 1;
S3312+B3422-S50B48 <= 1;

S3313+B3411-S51B45 <= 1;
S3313+B3412-S51B46 <= 1;
S3321+B3411-S52B45 <= 1;
S3321+B3412-S52B46 <= 1;
S3322+B3411-S53B45 <= 1;
S3322+B3412-S53B46 <= 1;
S3322+B3421-S53B47 <= 1;
S3322+B3422-S53B48 <= 1;
S3323+B3411-S54B45 <= 1;
S3323+B3412-S54B46 <= 1;
S4211+B4311-S61B57 <= 1;
S4211+B4312-S61B58 <= 1;
S4212+B4311-S62B57 <= 1;
S4212+B4312-S62B58 <= 1;
S4212+B4321-S62B59 <= 1;
S4212+B4322-S62B60 <= 1;
S4213+B4311-S63B57 <= 1;
S4213+B4312-S63B58 <= 1;
S1111+S1211+B1211-S1S7B5 <= 2;
S1111+S1211+B1212-S1S7B6 <= 2;
S1111+S1212+B1211-S1S8B5 <= 2;
S1111+S1212+B1212-S1S8B6 <= 2;
S1111+S1213+B1211-S1S9B5 <= 2;
S1111+S1213+B1212-S1S9B6 <= 2;
S1111+S1213+B1221-S1S9B7 <= 2;
S1111+S1213+B1222-S1S9B8 <= 2;
S1112+S1211+B1211-S2S7B5 <= 2;
S1112+S1211+B1212-S2S7B6 <= 2;
S1112+S1212+B1211-S2S8B5 <= 2;
S1112+S1212+B1212-S2S8B6 <= 2;
S1112+S1213+B1211-S2S9B5 <= 2;
S1112+S1213+B1212-S2S9B6 <= 2;
S1112+S1213+B1221-S2S9B7 <= 2;
S1112+S1213+B1222-S2S9B8 <= 2;
S1113+S1211+B1211-S3S7B5 <= 2;
S1113+S1211+B1212-S3S7B6 <= 2;
S1113+S1211+B1221-S3S7B7 <= 2;
S1113+S1211+B1222-S3S7B8 <= 2;
S1113+S1212+B1211-S3S8B5 <= 2;
S1113+S1212+B1212-S3S8B6 <= 2;
S1113+S1212+B1221-S3S8B7 <= 2;
S1113+S1212+B1222-S3S8B8 <= 2;
S1113+S1213+B1211-S3S9B5 <= 2;
S1113+S1213+B1212-S3S9B6 <= 2;
S1121+S1211+B1211-S4S7B5 <= 2;
S1121+S1211+B1212-S4S7B6 <= 2;
S1121+S1212+B1211-S4S8B5 <= 2;
S1121+S1212+B1212-S4S8B6 <= 2;
S1121+S1213+B1211-S4S9B5 <= 2;
S1121+S1213+B1212-S4S9B6 <= 2;
S1121+S1213+B1221-S4S9B7 <= 2;
S1121+S1213+B1222-S4S9B8 <= 2;
S1122+S1211+B1211-S5S7B5 <= 2;
S1122+S1211+B1212-S5S7B6 <= 2;
S1122+S1212+B1211-S5S8B5 <= 2;
S1122+S1212+B1212-S5S8B6 <= 2;
S1122+S1213+B1211-S5S9B5 <= 2;
S1122+S1213+B1212-S5S9B6 <= 2;
S1122+S1213+B1221-S5S9B7 <= 2;
S1122+S1213+B1222-S5S9B8 <= 2;
S1123+S1211+B1211-S6S7B5 <= 2;
S1123+S1211+B1212-S6S7B6 <= 2;
S1123+S1211+B1221-S6S7B7 <= 2;
S1123+S1211+B1222-S6S7B8 <= 2;
S1123+S1212+B1211-S6S8B5 <= 2;
S1123+S1212+B1212-S6S8B6 <= 2;
S1123+S1212+B1221-S6S8B7 <= 2;
S1123+S1212+B1222-S6S8B8 <= 2;
S1123+S1213+B1211-S6S9B5 <= 2;
S1123+S1213+B1212-S6S9B6 <= 2;
S1211+S1311+B1311-S7S13B9 <= 2;
S1211+S1311+B1312-S7S13B10 <= 2;
S1211+S1312+B1311-S7S14B9 <= 2;
S1211+S1312+B1312-S7S14B10 <= 2;
S1211+S1313+B1311-S7S15B9 <= 2;
S1211+S1313+B1312-S7S15B10 <= 2;
S1211+S1313+B1321-S7S15B11 <= 2;
S1211+S1313+B1322-S7S15B12 <= 2;
S1212+S1311+B1311-S8S13B9 <= 2;
S1212+S1311+B1312-S8S13B10 <= 2;
S1212+S1312+B1311-S8S14B9 <= 2;
S1212+S1312+B1312-S8S14B10 <= 2;
S1212+S1313+B1311-S8S15B9 <= 2;
S1212+S1313+B1312-S8S15B10 <= 2;
S1212+S1313+B1321-S8S15B11 <= 2;
S1212+S1313+B1322-S8S15B12 <= 2;
S1213+S1311+B1311-S9S13B9 <= 2;
S1213+S1311+B1312-S9S13B10 <= 2;
S1213+S1311+B1321-S9S13B11 <= 2;
S1213+S1311+B1322-S9S13B12 <= 2;
S1213+S1312+B1311-S9S14B9 <= 2;
S1213+S1312+B1312-S9S14B10 <= 2;
S1213+S1312+B1321-S9S14B11 <= 2;
S1213+S1312+B1322-S9S14B12 <= 2;
S1213+S1313+B1311-S9S15B9 <= 2;
S1213+S1313+B1312-S9S15B10 <= 2;
S1211+S1321+B1311-S7S16B9 <= 2;
S1211+S1321+B1312-S7S16B10 <= 2;
S1211+S1322+B1311-S7S17B9 <= 2;
S1211+S1322+B1312-S7S17B10 <= 2;
S1211+S1323+B1311-S7S18B9 <= 2;
S1211+S1323+B1312-S7S18B10 <= 2;
S1211+S1323+B1321-S7S18B11 <= 2;
S1211+S1323+B1322-S7S18B12 <= 2;
S1212+S1321+B1311-S8S16B9 <= 2;
S1212+S1321+B1312-S8S16B10 <= 2;
S1212+S1322+B1311-S8S17B9 <= 2;
S1212+S1322+B1312-S8S17B10 <= 2;
S1212+S1323+B1311-S8S18B9 <= 2;
S1212+S1323+B1312-S8S18B10 <= 2;
S1212+S1323+B1321-S8S18B11 <= 2;
S1212+S1323+B1322-S8S18B12 <= 2;
S1213+S1321+B1311-S9S16B9 <= 2;
S1213+S1321+B1312-S9S16B10 <= 2;
S1213+S1321+B1321-S9S16B11 <= 2;
S1213+S1321+B1322-S9S16B12 <= 2;
S1213+S1322+B1311-S9S17B9 <= 2;
S1213+S1322+B1312-S9S17B10 <= 2;
S1213+S1322+B1321-S9S17B11 <= 2;
S1213+S1322+B1322-S9S17B12 <= 2;
S1213+S1323+B1311-S9S18B9 <= 2;
S1213+S1323+B1312-S9S18B10 <= 2;
S2111+S2211+B2211-S19S25B21 <= 2;
S2111+S2211+B2212-S19S25B22 <= 2;
S2111+S2212+B2211-S19S26B21 <= 2;
S2111+S2212+B2212-S19S26B22 <= 2;
S2111+S2213+B2211-S19S27B21 <= 2;
S2111+S2213+B2212-S19S27B22 <= 2;
S2111+S2213+B2221-S19S27B23 <= 2;
S2111+S2213+B2222-S19S27B24 <= 2;
S2112+S2211+B2211-S20S25B21 <= 2;
S2112+S2211+B2212-S20S25B22 <= 2;
S2112+S2212+B2211-S20S26B21 <= 2;
S2112+S2212+B2212-S20S26B22 <= 2;
S2112+S2213+B2211-S20S27B21 <= 2;
S2112+S2213+B2212-S20S27B22 <= 2;
S2112+S2213+B2221-S20S27B23 <= 2;
S2112+S2213+B2222-S20S27B24 <= 2;
S2113+S2211+B2211-S21S25B21 <= 2;
S2113+S2211+B2212-S21S25B22 <= 2;
S2113+S2211+B2221-S21S25B23 <= 2;
S2113+S2211+B2222-S21S25B24 <= 2;
S2113+S2212+B2211-S21S26B21 <= 2;
S2113+S2212+B2212-S21S26B22 <= 2;
S2113+S2212+B2221-S21S26B23 <= 2;
S2113+S2212+B2222-S21S26B24 <= 2;
S2113+S2213+B2211-S21S27B21 <= 2;
S2113+S2213+B2212-S21S27B22 <= 2;
S2111+S2221+B2211-S19S28B21 <= 2;
S2111+S2221+B2212-S19S28B22 <= 2;
S2111+S2222+B2211-S19S29B21 <= 2;
S2111+S2222+B2212-S19S29B22 <= 2;
S2111+S2223+B2211-S19S30B21 <= 2;
S2111+S2223+B2212-S19S30B22 <= 2;
S2111+S2223+B2221-S19S30B23 <= 2;
S2111+S2223+B2222-S19S30B24 <= 2;
S2112+S2221+B2211-S20S28B21 <= 2;
S2112+S2221+B2212-S20S28B22 <= 2;
S2112+S2222+B2211-S20S29B21 <= 2;
S2112+S2222+B2212-S20S29B22 <= 2;
S2112+S2223+B2211-S20S30B21 <= 2;
S2112+S2223+B2212-S20S30B22 <= 2;
S2112+S2223+B2221-S20S30B23 <= 2;
S2112+S2223+B2222-S20S30B24 <= 2;
S2113+S2221+B2211-S21S28B21 <= 2;
S2113+S2221+B2212-S21S28B22 <= 2;
S2113+S2221+B2221-S21S28B23 <= 2;
S2113+S2221+B2222-S21S28B24 <= 2;
S2113+S2222+B2211-S21S29B21 <= 2;
S2113+S2222+B2212-S21S29B22 <= 2;
S2113+S2222+B2221-S21S29B23 <= 2;
S2113+S2222+B2222-S21S29B24 <= 2;
S2113+S2223+B2211-S21S30B21 <= 2;
S2113+S2223+B2212-S21S30B22 <= 2;
S2121+S2211+B2211-S22S25B21 <= 2;
S2121+S2211+B2212-S22S25B22 <= 2;
S2121+S2212+B2211-S22S26B21 <= 2;
S2121+S2212+B2212-S22S26B22 <= 2;
S2121+S2213+B2211-S22S27B21 <= 2;
S2121+S2213+B2212-S22S27B22 <= 2;
S2121+S2213+B2221-S22S27B23 <= 2;
S2121+S2213+B2222-S22S27B24 <= 2;
S2122+S2211+B2211-S23S25B21 <= 2;
S2122+S2211+B2212-S23S25B22 <= 2;
S2122+S2212+B2211-S23S26B21 <= 2;
S2122+S2212+B2212-S23S26B22 <= 2;
S2122+S2213+B2211-S23S27B21 <= 2;
S2122+S2213+B2212-S23S27B22 <= 2;
S2122+S2213+B2221-S23S27B23 <= 2;
S2122+S2213+B2222-S23S27B24 <= 2;
S2123+S2211+B2211-S24S25B21 <= 2;
S2123+S2211+B2212-S24S25B22 <= 2;
S2123+S2211+B2221-S24S25B23 <= 2;
S2123+S2211+B2222-S24S25B24 <= 2;
S2123+S2212+B2211-S24S26B21 <= 2;
S2123+S2212+B2212-S24S26B22 <= 2;
S2123+S2212+B2221-S24S26B23 <= 2;
S2123+S2212+B2222-S24S26B24 <= 2;
S2123+S2213+B2211-S24S27B21 <= 2;
S2123+S2213+B2212-S24S27B22 <= 2;
S2121+S2221+B2211-S22S28B21 <= 2;
S2121+S2221+B2212-S22S28B22 <= 2;
S2121+S2222+B2211-S22S29B21 <= 2;
S2121+S2222+B2212-S22S29B22 <= 2;
S2121+S2223+B2211-S22S30B21 <= 2;
S2121+S2223+B2212-S22S30B22 <= 2;
S2121+S2223+B2221-S22S30B23 <= 2;
S2121+S2223+B2222-S22S30B24 <= 2;
S2122+S2221+B2211-S23S28B21 <= 2;
S2122+S2221+B2212-S23S28B22 <= 2;

S2122+S2222+B2211-S23S29B21 <= 2; S2122+S2222+B2212-S23S29B22 <= 2; S2122+S2223+B2211-S23S30B21 <= 2;
S2122+S2223+B2212-S23S30B22 <= 2; S2122+S2223+B2221-S23S30B23 <= 2; S2122+S2223+B2222-S23S30B24 <= 2;
S2123+S2221+B2211-S24S28B21 <= 2; S2123+S2221+B2212-S24S28B22 <= 2; S2123+S2221+B2221-S24S28B23 <= 2;
S2123+S2221+B2222-S24S28B24 <= 2; S2123+S2222+B2211-S24S29B21 <= 2; S2123+S2222+B2212-S24S29B22 <= 2;
S2123+S2222+B2221-S24S29B23 <= 2; S2123+S2222+B2222-S24S29B24 <= 2; S2123+S2223+B2211-S24S30B21 <= 2;
S2123+S2223+B2212-S24S30B22 <= 2; S3111+S3211+B3211-S37S43B37 <= 2; S3111+S3211+B3212-S37S43B38 <= 2;
S3111+S3212+B3211-S37S44B37 <= 2; S3111+S3212+B3212-S37S44B38 <= 2; S3111+S3213+B3211-S37S45B37 <= 2;
S3111+S3213+B3212-S37S45B38 <= 2; S3111+S3213+B3221-S37S45B39 <= 2; S3111+S3213+B3222-S37S45B40 <= 2;
S3112+S3211+B3211-S38S43B37 <= 2; S3112+S3211+B3212-S38S43B38 <= 2; S3112+S3212+B3211-S38S44B37 <= 2;
S3112+S3212+B3212-S38S44B38 <= 2; S3112+S3213+B3211-S38S45B37 <= 2; S3112+S3213+B3212-S38S45B38 <= 2;
S3112+S3213+B3221-S38S45B39 <= 2; S3112+S3213+B3222-S38S45B40 <= 2; S3113+S3211+B3211-S39S43B37 <= 2;
S3113+S3211+B3212-S39S43B38 <= 2; S3113+S3211+B3221-S39S43B39 <= 2; S3113+S3211+B3222-S39S43B40 <= 2;
S3113+S3212+B3211-S39S44B37 <= 2; S3113+S3212+B3212-S39S44B38 <= 2; S3113+S3212+B3221-S39S44B39 <= 2;
S3113+S3212+B3222-S39S44B40 <= 2; S3113+S3213+B3211-S39S45B37 <= 2; S3113+S3213+B3212-S39S45B38 <= 2;
S3121+S3211+B3211-S40S43B37 <= 2; S3121+S3211+B3212-S40S43B38 <= 2; S3121+S3212+B3211-S40S44B37 <= 2;
S3121+S3212+B3212-S40S44B38 <= 2; S3121+S3213+B3211-S40S45B37 <= 2; S3121+S3213+B3212-S40S45B38 <= 2;
S3121+S3213+B3221-S40S45B39 <= 2; S3121+S3213+B3222-S40S45B40 <= 2; S3122+S3211+B3211-S41S43B37 <= 2;
S3122+S3211+B3212-S41S43B38 <= 2; S3122+S3212+B3211-S41S44B37 <= 2; S3122+S3212+B3212-S41S44B38 <= 2;
S3122+S3213+B3211-S41S45B37 <= 2; S3122+S3213+B3212-S41S45B38 <= 2; S3122+S3213+B3221-S41S45B39 <= 2;
S3122+S3213+B3222-S41S45B40 <= 2; S3123+S3211+B3211-S42S43B37 <= 2; S3123+S3211+B3212-S42S43B38 <= 2;
S3123+S3211+B3221-S42S43B39 <= 2; S3123+S3211+B3222-S42S43B40 <= 2; S3123+S3212+B3211-S42S44B37 <= 2;
S3123+S3212+B3212-S42S44B38 <= 2; S3123+S3212+B3221-S42S44B39 <= 2; S3123+S3212+B3222-S42S44B40 <= 2;
S3123+S3213+B3211-S42S45B37 <= 2; S3123+S3213+B3212-S42S45B38 <= 2; S3211+S3311+B3311-S43S49B41 <= 2;
S3211+S3311+B3312-S43S49B42 <= 2; S3211+S3312+B3311-S43S50B41 <= 2; S3211+S3312+B3312-S43S50B42 <= 2;
S3211+S3313+B3311-S43S51B41 <= 2; S3211+S3313+B3312-S43S51B42 <= 2; S3211+S3313+B3321-S43S51B43 <= 2;
S3211+S3313+B3322-S43S51B44 <= 2; S3212+S3311+B3311-S44S49B41 <= 2; S3212+S3311+B3312-S44S49B42 <= 2;
S3212+S3312+B3311-S44S50B41 <= 2; S3212+S3312+B3312-S44S50B42 <= 2; S3212+S3313+B3311-S44S51B41 <= 2;
S3212+S3313+B3312-S44S51B42 <= 2; S3212+S3313+B3321-S44S51B43 <= 2; S3212+S3313+B3322-S44S51B44 <= 2;
S3213+S3311+B3311-S45S49B41 <= 2; S3213+S3311+B3312-S45S49B42 <= 2; S3213+S3311+B3321-S45S49B43 <= 2;
S3213+S3311+B3322-S45S49B44 <= 2; S3213+S3312+B3311-S45S50B41 <= 2; S3213+S3312+B3312-S45S50B42 <= 2;
S3213+S3312+B3321-S45S50B43 <= 2; S3213+S3312+B3322-S45S50B44 <= 2; S3213+S3313+B3311-S45S51B41 <= 2;
S3213+S3313+B3312-S45S51B42 <= 2; S3211+S3321+B3311-S43S52B41 <= 2; S3211+S3321+B3312-S43S52B42 <= 2;
S3211+S3322+B3311-S43S53B41 <= 2; S3211+S3322+B3312-S43S53B42 <= 2; S3211+S3323+B3311-S43S54B41 <= 2;
S3211+S3323+B3312-S43S54B42 <= 2; S3211+S3323+B3321-S43S54B43 <= 2; S3211+S3323+B3322-S43S54B44 <= 2;
S3212+S3321+B3311-S44S52B41 <= 2; S3212+S3321+B3312-S44S52B42 <= 2; S3212+S3322+B3311-S44S53B41 <= 2;
S3212+S3322+B3312-S44S53B42 <= 2; S3212+S3323+B3311-S44S54B41 <= 2; S3212+S3323+B3312-S44S54B42 <= 2;
S3212+S3323+B3321-S44S54B43 <= 2; S3212+S3323+B3322-S44S54B44 <= 2; S3213+S3321+B3311-S45S52B41 <= 2;
S3213+S3321+B3312-S45S52B42 <= 2; S3213+S3321+B3321-S45S52B43 <= 2; S3213+S3321+B3322-S45S52B44 <= 2;
S3213+S3322+B3311-S45S53B41 <= 2; S3213+S3322+B3312-S45S53B42 <= 2; S3213+S3322+B3321-S45S53B43 <= 2;
S3213+S3322+B3322-S45S53B44 <= 2; S3213+S3323+B3311-S45S54B41 <= 2; S3213+S3323+B3312-S45S54B42 <= 2;
S4111+S4211+B4211-S55S61B53 <= 2; S4111+S4211+B4212-S55S61B54 <= 2; S4111+S4212+B4211-S55S62B53 <= 2;
S4111+S4212+B4212-S55S62B54 <= 2; S4111+S4213+B4211-S55S63B53 <= 2; S4111+S4213+B4212-S55S63B54 <= 2;
S4111+S4213+B4221-S55S63B55 <= 2; S4111+S4213+B4222-S55S63B56 <= 2; S4112+S4211+B4211-S56S61B53 <= 2;
S4112+S4211+B4212-S56S61B54 <= 2; S4112+S4212+B4211-S56S62B53 <= 2; S4112+S4212+B4212-S56S62B54 <= 2;
S4112+S4213+B4211-S56S63B53 <= 2; S4112+S4213+B4212-S56S63B54 <= 2; S4112+S4213+B4221-S56S63B55 <= 2;
S4112+S4213+B4222-S56S63B56 <= 2; S4113+S4211+B4211-S57S61B53 <= 2; S4113+S4211+B4212-S57S61B54 <= 2;
S4113+S4211+B4221-S57S61B55 <= 2; S4113+S4211+B4222-S57S61B56 <= 2; S4113+S4212+B4211-S57S62B53 <= 2;
S4113+S4212+B4212-S57S62B54 <= 2; S4113+S4212+B4221-S57S62B55 <= 2; S4113+S4212+B4222-S57S62B56 <= 2;
S4113+S4213+B4211-S57S63B53 <= 2; S4113+S4213+B4212-S57S63B54 <= 2; S4121+S4211+B4211-S58S61B53 <= 2;
S4121+S4211+B4212-S58S61B54 <= 2; S4121+S4212+B4211-S58S62B53 <= 2; S4121+S4212+B4212-S58S62B54 <= 2;
S4121+S4213+B4211-S58S63B53 <= 2; S4121+S4213+B4212-S58S63B54 <= 2; S4121+S4213+B4221-S58S63B55 <= 2;
S4121+S4213+B4222-S58S63B56 <= 2; S4122+S4211+B4211-S59S61B53 <= 2; S4122+S4211+B4212-S59S61B54 <= 2;
S4122+S4212+B4211-S59S62B53 <= 2; S4122+S4212+B4212-S59S62B54 <= 2; S4122+S4213+B4211-S59S63B53 <= 2;
S4122+S4213+B4212-S59S63B54 <= 2; S4122+S4213+B4221-S59S63B55 <= 2; S4122+S4213+B4222-S59S63B56 <= 2;
S4123+S4211+B4211-S60S61B53 <= 2; S4123+S4211+B4212-S60S61B54 <= 2; S4123+S4211+B4221-S60S61B55 <= 2;
S4123+S4211+B4222-S60S61B56 <= 2; S4123+S4212+B4211-S60S62B53 <= 2; S4123+S4212+B4212-S60S62B54 <= 2;
S4123+S4212+B4221-S60S62B55 <= 2; S4123+S4212+B4222-S60S62B56 <= 2; S4123+S4213+B4211-S60S63B53 <= 2;
S4123+S4213+B4212-S60S63B54 <= 2;

!These are the second constraints imposed by the transformation proposed by Glover and Woolsey (1972). Refer to Chapter 6 for detailed explanation of these constraints. It extracts the SB variables that are related to specific S variable;

S1B1+S1B2+S1S7B5+S1S7B6+S1S8B5+S1S8B6+S1S9B5+S1S9B6+S1S9B7+S1S9B8 - 10*S1111 <= 0;
S2B1+S2B2+S2B3+S2B4+S2S7B5+S2S7B6+S2S8B5+S2S8B6+S2S9B5+S2S9B6+S2S9B7+S2S9B8 - 12*S1112 <= 0;
S3B1+S3B2+S3S7B5+S3S7B6+S3S7B7+S3S7B8+S3S8B5+S3S8B6+S3S8B7+S3S8B8+S3S9B5+S3S9B6 - 12*S1113 <= 0;
S4B1+S4B2+S4S7B5+S4S7B6+S4S8B5+S4S8B6+S4S9B5+S4S9B6+S4S9B7+S4S9B8 - 10*S1121 <= 0;
S5B1+S5B2+S5B3+S5B4+S5S7B5+S5S7B6+S5S8B5+S5S8B6+S5S9B5+S5S9B6+S5S9B7+S5S9B8 -12*S1122 <= 0;
S6B1+S6B2+S6S7B5+S6S7B6+S6S7B7+S6S7B8+S6S8B5+S6S8B6+S6S8B7+S6S8B8+S6S9B5+S6S9B6 - 12*S1123 <= 0;
S1S7B5+S1S7B6+S2S7B5+S2S7B6+S3S7B5+S3S7B6+S3S7B7+S3S7B8+S4S7B5+S4S7B6+S5S7B5+S5S7B6+S6S7B5+S6S7B6+
S6S7B7+S6S7B8+S7S13B9+S7S13B10+S7S14B9+S7S14B10+S7S15B9+S7S15B10+S7S15B11+S7S15B12+S7S16B9+

S7S16B10+S7S17B9+S7S17B10+S7S18B9+S7S18B10+S7S18B11+S7S18B12 - 32*S1211 <= 0;
S1S8B5+S1S8B6+S2S8B5+S2S8B6+S3S8B5+S3S8B6+S3S8B7+S3S8B8+S4S8B5+S4S8B6+S5S8B5+S5S8B6+S6S8B5+S6S8B6+
S6S8B7+S6S8B8+S8S13B9+S8S13B10+S8S14B9+S8S14B10+S8S15B9+S8S15B10+S8S15B11+S8S15B12+S8S16B9+
S8S16B10+S8S17B9+S8S17B10+S8S18B9+S8S18B10+S8S18B11+S8S18B12 - 32*S1212 <= 0;
S1S9B5+S1S9B6+S1S9B7+S1S9B8+S2S9B5+S2S9B6+S2S9B7+S2S9B8+S3S9B5+S3S9B6+S4S9B5+S4S9B6+S4S9B7+S4S9B8+
S5S9B5+S5S9B6+S5S9B7+S5S9B8+S6S9B5+S6S9B6+S9S13B9+S9S13B10+S9S13B11+S9S13B12+S9S14B9+S9S14B10+
S9S14B11+S9S14B12+S9S15B9+S9S15B10+S9S16B9+S9S16B10+S9S16B11+S9S16B12+S9S17B9+S9S17B10+S9S17B11+
S9S17B12+S9S18B9+S9S18B10 - 40*S1213 <= 0;
S13B13+S13B14+S7S13B9+S7S13B10+S8S13B9+S8S13B10+S9S13B9+S9S13B10+S9S13B11+S9S13B12 - 10*S1311 <= 0;
S14B13+S14B14+S14B15+S14B16+S7S14B9+S7S14B10+S8S14B9+S8S14B10+S9S14B9+S9S14B10+S9S14B11+S9S14B12 –
12*S1312 <= 0;
S15B13+S15B14+S7S15B9+S7S15B10+S7S15B11+S7S15B12+S8S15B9+S8S15B10+S8S15B11+S8S15B12+S9S15B9+
S9S15B10 – 12*S1313 <= 0;
S16B13+S16B14+S7S16B9+S7S16B10+S8S16B9+S8S16B10+S9S16B9+S9S16B10+S9S16B11+S9S16B12 - 10*S1321 <= 0;
S17B13+S17B14+S17B15+S17B16+S7S17B9+S7S17B10+S8S17B9+S8S17B10+S9S17B9+S9S17B10+S9S17B11+S9S17B12 –
12*S1322 <= 0;
S18B13+S18B14+S7S18B9+S7S18B10+S7S18B11+S7S18B12+S8S18B9+S8S18B10+S8S18B11+S8S18B12+S9S18B9+
S9S18B10– 12*S1323 <= 0;
S19B17+S19B18+S19S25B21+S19S25B22+S19S26B21+S19S26B22+S19S27B21+S19S27B22+S19S27B23+S19S27B24+
S19S28B21+S19S28B22+S19S29B21+S19S29B22+S19S30B21+S19S30B22+S19S30B23+S19S30B24 - 18*S2111 <= 0;
S20B17+S20B18+S20B19+S20B20+S20S25B21+S20S25B22+S20S26B21+S20S26B22+S20S27B21+S20S27B22+S20S27B23+
S20S27B24+S20S28B21+S20S28B22+S20S29B21+S20S29B22+S20S30B21+S20S30B22+S20S30B23+S20S30B24 –
20*S2112 <= 0;
S21B17+S21B18+S21S25B21+S21S25B22+S21S25B23+S21S25B24+S21S26B21+S21S26B22+S21S26B23+S21S26B24+
S21S27B21+S21S27B22+S21S28B21+S21S28B22+S21S28B23+S21S28B24+S21S29B21+S21S29B22+S21S29B23+S21S29B24+
S21S30B21+S21S30B22 - 22*S2113 <= 0;
S22B17+S22B18+S22S25B21+S22S25B22+S22S26B21+S22S26B22+S22S27B21+S22S27B22+S22S27B23+S22S27B24+
S22S28B21+S22S28B22+S22S29B21+S22S29B22+S22S30B21+S22S30B22+S22S30B23+S22S30B24 - 18*S2121 <= 0;
S23B17+S23B18+S23B19+S23B20+S23S25B21+S23S25B22+S23S26B21+S23S26B22+S23S27B21+S23S27B22+S23S27B23+
S23S27B24+S23S28B21+S23S28B22+S23S29B21+S23S29B22+S23S30B21+S23S30B22+S23S30B23+S23S30B24 –
20*S2122 <= 0;
S24B17+S24B18+S24S25B21+S24S25B22+S24S25B23+S24S25B24+S24S26B21+S24S26B22+S24S26B23+S24S26B24+
S24S27B21+S24S27B22+S24S28B21+S24S28B22+S24S28B23+S24S28B24+S24S29B21+S24S29B22+S24S29B23+S24S29B24+
S24S30B21+S24S30B22 - 22*S2123 <= 0;
S25B25+S25B26+S19S25B21+S19S25B22+S20S25B21+S20S25B22+S21S25B21+S21S25B22+S21S25B23+S21S25B24+
S22S25B21+S22S25B22+S23S25B21+S23S25B22+S24S25B21+S24S25B22+S24S25B23+S24S25B24 - 18*S2211 <= 0;
S26B25+S26B26+S26B27+S26B28+S19S26B21+S19S26B22+S20S26B21+S20S26B22+S21S26B21+S21S26B22+S21S26B23+
S21S26B24+S22S26B21+S22S26B22+S23S26B21+S23S26B22+S24S26B21+S24S26B22+S24S26B23+S24S26B24 –
20*S2212 <= 0;
S27B25+S27B26+S19S27B21+S19S27B22+S19S27B23+S19S27B24+S20S27B21+S20S27B22+S20S27B23+S20S27B24+
S21S27B21+S21S27B22+S22S27B21+S22S27B22+S22S27B23+S22S27B24+S23S27B21+S23S27B22+S23S27B23+S23S27B24+
S24S27B21+S24S27B22 - 22*S2213 <= 0;
S28B25+S28B26+S19S28B21+S19S28B22+S20S28B21+S20S28B22+S21S28B21+S21S28B22+S21S28B23+S21S28B24+
S22S28B21+S22S28B22+S23S28B21+S23S28B22+S24S28B21+S24S28B22+S24S28B23+S24S28B24 - 18*S2221 <= 0;
S29B25+S29B26+S29B27+S29B28+S19S29B21+S19S29B22+S20S29B21+S20S29B22+S21S29B21+S21S29B22+S21S29B23+
S21S29B24+S22S29B21+S22S29B22+S23S29B21+S23S29B22+S24S29B21+S24S29B22+S24S29B23+S24S29B24 –
20*S2222 <= 0;
S30B25+S30B26+S19S30B21+S19S30B22+S19S30B23+S19S30B24+S20S30B21+S20S30B22+S20S30B23+S20S30B24+
S21S30B21+S21S30B22+S22S30B21+S22S30B22+S22S30B23+S22S30B24+S23S30B21+S23S30B22+S23S30B23+S23S30B24+
S24S30B21+S24S30B22 - 22*S2223 <= 0;
S37B33+S37B34+S37S43B37+S37S43B38+S37S44B37+S37S44B38+S37S45B37+S37S45B38+S37S45B39+S37S45B40 -
10*S3111 <= 0;
S38B33+S38B34+S38B35+S38B36+S38S43B37+S38S43B38+S38S44B37+S38S44B38+S38S45B37+S38S45B38+S38S45B39+
S38S45B40 - 12*S3112 <= 0;
S39B33+S39B34+S39S43B37+S39S43B38+S39S43B39+S39S43B40+S39S44B37+S39S44B38+S39S44B39+S39S44B40+
S39S45B37+S39S45B38 - 12*S3113 <= 0;
S40B33+S40B34+S40S43B37+S40S43B38+S40S44B37+S40S44B38+S40S45B37+S40S45B38+S40S45B39+S40S45B40 -
10*S3121 <= 0;
S41B33+S41B34+S41B35+S41B36+S41S43B37+S41S43B38+S41S44B37+S41S44B38+S41S45B37+S41S45B38+S41S45B39+
S41S45B40 - 12*S3122 <= 0;
S42B33+S42B34+S42S43B37+S42S43B38+S42S43B39+S42S43B40+S42S44B37+S42S44B38+S42S44B39+S42S44B40+
S42S45B37+S42S45B38 - 12*S3123 <= 0;
S37S43B37+S37S43B38+S38S43B37+S38S43B38+S39S43B37+S39S43B38+S39S43B39+S39S43B40+S40S43B37+S40S43B38+
S41S43B37+S41S43B38+S42S43B37+S42S43B38+S42S43B39+S42S43B40+S43S49B41+S43S49B42+S43S50B41+S43S50B42+
S43S51B41+S43S51B42+S43S51B43+S43S51B44+S43S52B41+S43S52B42+S43S53B41+S43S53B42+S43S54B41+S43S54B42+
S43S54B43+S43S54B44 - 32*S3211 <= 0;
S37S44B37+S37S44B38+S38S44B37+S38S44B38+S39S44B37+S39S44B38+S39S44B39+S39S44B40+S40S44B37+S40S44B38+
S41S44B37+S41S44B38+S42S44B37+S42S44B38+S42S44B39+S42S44B40+S44S49B41+S44S49B42+S44S50B41+S44S50B42+
S44S51B41+S44S51B42+S44S51B43+S44S51B44+S44S52B41+S44S52B42+S44S53B41+S44S53B42+S44S54B41+S44S54B42+

S44S54B43+S44S54B44 - 32*S3212 <= 0;
S37S45B37+S37S45B38+S37S45B39+S37S45B40+S38S45B37+S38S45B38+S38S45B39+S38S45B40+S39S45B37+S39S45B38+
S40S45B37+S40S45B38+S40S45B39+S40S45B40+S41S45B37+S41S45B38+S41S45B39+S41S45B40+S42S45B37+S42S45B38+
S45S49B41+S45S49B42+S45S49B43+S45S49B44+S45S50B41+S45S50B42+S45S50B43+S45S50B44+S45S51B41+S45S51B42+
S45S52B41+S45S52B42+S45S52B43+S45S52B44+S45S53B41+S45S53B42+S45S53B43+S45S53B44+S45S54B41+S45S54B42 –
40*S3213 <= 0;
S49B45+S49B46+S43S49B41+S43S49B42+S44S49B41+S44S49B42+S45S49B41+S45S49B42+S45S49B43+S45S49B44 -
10*S3311 <= 0;
S50B45+S50B46+S50B47+S50B48+S43S50B41+S43S50B42+S44S50B41+S44S50B42+S45S50B41+S45S50B42+S45S50B43+
S45S50B44 - 12*S3312 <= 0;
S51B45+S51B46+S43S51B41+S43S51B42+S43S51B43+S43S51B44+S44S51B41+S44S51B42+S44S51B43+S44S51B44+
S45S51B41+S45S51B42 - 12*S3313 <= 0;
S52B45+S52B46+S43S52B41+S43S52B42+S44S52B41+S44S52B42+S45S52B41+S45S52B42+S45S52B43+S45S52B44 -
10*S3321 <= 0;
S53B45+S53B46+S53B47+S53B48+S43S53B41+S43S53B42+S44S53B41+S44S53B42+S45S53B41+S45S53B42+S45S53B43+
S45S53B44 - 12*S3322 <= 0;
S54B45+S54B46+S43S54B41+S43S54B42+S43S54B43+S43S54B44+S44S54B41+S44S54B42+S44S54B43+S44S54B44+
S45S54B41+S45S54B42 -12*S3323 <= 0;
S55B49+S55B50+S55S61B53+S55S61B54+S55S62B53+S55S62B54+S55S63B53+S55S63B54+S55S63B55+S55S63B56 -
10*S4111 <= 0;
S56B49+S56B50+S56B51+S56B52+S56S61B53+S56S61B54+S56S62B53+S56S62B54+S56S63B53+S56S63B54+S56S63B55+
S56S63B56 - 12*S4112 <= 0;
S57B49+S57B50+S57S61B53+S57S61B54+S57S61B55+S57S61B56+S57S62B53+S57S62B54+S57S62B55+S57S62B56+
S57S63B53+S57S63B54 - 12*S4113 <= 0;
S58B49+S58B50+S58S61B53+S58S61B54+S58S62B53+S58S62B54+S58S63B53+S58S63B54+S58S63B55+S58S63B56 -
10*S4121 <= 0;
S59B49+S59B50+S59B51+S59B52+S59S61B53+S59S61B54+S59S62B53+S59S62B54+S59S63B53+S59S63B54+S59S63B55+
S59S63B56 - 12*S4122 <= 0;
S60B49+S60B50+S60S61B53+S60S61B54+S60S61B55+S60S61B56+S60S62B53+S60S62B54+S60S62B55+S60S62B56+
S60S63B53+S60S63B54 - 12*S4123 <= 0;
S61B57+S61B58+S55S61B53+S55S61B54+S56S61B53+S56S61B54+S57S61B53+S57S61B54+S57S61B55+S57S61B56+
S58S61B53+S58S61B54+S59S61B53+S59S61B54+S60S61B53+S60S61B54+S60S61B55+S60S61B56 - 18*S4211 <= 0;
S62B57+S62B58+S62B59+S62B60+S55S62B53+S55S62B54+S56S62B53+S56S62B54+S57S62B53+S57S62B54+S57S62B55+
S57S62B56+S58S62B53+S58S62B54+S59S62B53+S59S62B54+S60S62B53+S60S62B54+S60S62B55+S60S62B56 –
20*S4212 <= 0;
S63B57+S63B58+S55S63B53+S55S63B54+S55S63B55+S55S63B56+S56S63B53+S56S63B54+S56S63B55+S56S63B56+
S57S63B53+S57S63B54+S58S63B53+S58S63B54+S58S63B55+S58S63B56+S59S63B53+S59S63B54+S59S63B55+S59S63B56+
S60S63B53+S60S63B54 - 22*S4213 <= 0;

!These are the second constraints imposed by the transformation proposed by Glover and Woolsey (1972). Refer to Chapter 6 for detailed explanation of these constraints. It extracts the SB variables that are related to specific B variable;

S1B1+S2B1+S3B1+S4B1+S5B1+S6B1 - 6*B1111 <= 0;
S1B2+S2B2+S3B2+S4B2+S5B2+S6B2 - 6*B1112 <= 0;
S2B3+S5B3 - 2*B1121 <= 0;
S2B4+S5B4 - 2*B1122 <= 0;
S1S7B5+S1S8B5+S1S9B5+S2S7B5+S2S8B5+S2S9B5+S3S7B5+S3S8B5+S3S9B5+S4S7B5+S4S8B5+S4S9B5+S5S7B5+S5S8B5+
S5S9B5+S6S7B5+S6S8B5+S6S9B5 - 18*B1211 <= 0;
S1S7B6+S1S8B6+S1S9B6+S2S7B6+S2S8B6+S2S9B6+S3S7B6+S3S8B6+S3S9B6+S4S7B6+S4S8B6+S4S9B6+S5S7B6+S5S8B6+
S5S9B6+S6S7B6+S6S8B6+S6S9B6 - 18*B1212 <= 0;
S1S9B7+S2S9B7+S3S7B7+S3S8B7+S4S9B7+S5S9B7+S6S7B7+S6S8B7 - 8*B1221 <= 0;
S1S9B8+S2S9B8+S3S7B8+S3S8B8+S4S9B8+S5S9B8+S6S7B8+S6S8B8 - 8*B1222 <= 0;
S7S13B9+S7S14B9+S7S15B9+S8S13B9+S8S14B9+S8S15B9+S9S13B9+S9S14B9+S9S15B9+S7S16B9+S7S17B9+S7S18B9+
S8S16B9+S8S17B9+S8S18B9+S9S16B9+S9S17B9+S9S18B9 -18*B1311 <= 0;
S7S13B10+S7S14B10+S7S15B10+S8S13B10+S8S14B10+S8S15B10+S9S13B10+S9S14B10+S9S15B10+S7S16B10+S7S17B10+
S7S18B10+S8S16B10+S8S17B10+S8S18B10+S9S16B10+S9S17B10+S9S18B10 - 18*B1312 <= 0;
S7S15B11+S8S15B11+S9S13B11+S9S14B11+S7S18B11+S8S18B11+S9S16B11+S9S17B11 - 8*B1321 <= 0;
S7S15B12+S8S15B12+S9S13B12+S9S14B12+S7S18B12+S8S18B12+S9S16B12+S9S17B12 - 8*B1322 <= 0;
S13B13+S14B13+S15B13+S16B13+S17B13+S18B13 - 6*B1411 <= 0;
S13B14+S14B14+S15B14+S16B14+S17B14+S18B14 - 6*B1412 <= 0;
S14B15+S17B15 - 2*B1421 <= 0;
S14B16+S17B16 - 2*B1422 <= 0;
S19B17+S20B17+S21B17+S22B17+S23B17+S24B17 - 6*B2111 <= 0;
S19B18+S20B18+S21B18+S22B18+S23B18+S24B18 - 6*B2112 <= 0;
S20B19+S23B19 - 2*B2121 <= 0;
S20B20+S23B20 - 2*B2122 <= 0;
S19S25B21+S19S26B21+S19S27B21+S20S25B21+S20S26B21+S20S27B21+S21S25B21+S21S26B21+S21S27B21+S19S28B21+
S19S29B21+S19S30B21+S20S28B21+S20S29B21+S20S30B21+S21S28B21+S21S29B21+S21S30B21+S22S25B21+S22S26B21+
S22S27B21+S23S25B21+S23S26B21+S23S27B21+S24S25B21+S24S26B21+S24S27B21+S22S28B21+S22S29B21+S22S30B21+

S23S28B21+S23S29B21+S23S30B21+S24S28B21+S24S29B21+S24S30B21 - 36*B2211 <= 0;
S19S25B22+S19S26B22+S19S27B22+S20S25B22+S20S26B22+S20S27B22+S21S25B22+S21S26B22+S21S27B22+S19S28B22+
S19S29B22+S19S30B22+S20S28B22+S20S29B22+S20S30B22+S21S28B22+S21S29B22+S21S30B22+S22S25B22+S22S26B22+
S22S27B22+S23S25B22+S23S26B22+S23S27B22+S24S25B22+S24S26B22+S24S27B22+S22S28B22+S22S29B22+S22S30B22+
S23S28B22+S23S29B22+S23S30B22+S24S28B22+S24S29B22+S24S30B22 - 36*B2212 <= 0;
S19S27B23+S20S27B23+S21S25B23+S21S26B23+S19S30B23+S20S30B23+S21S28B23+S21S29B23+S22S27B23+S23S27B23+
S24S25B23+S24S26B23+S22S30B23+S23S30B23+S24S28B23+S24S29B23 - 16*B2221 <= 0;
S19S27B24+S20S27B24+S21S25B24+S21S26B24+S19S30B24+S20S30B24+S21S28B24+S21S29B24+S22S27B24+S23S27B24+
S24S25B24+S24S26B24+S22S30B24+S23S30B24+S24S28B24+S24S29B24 - 16*B2222 <= 0;
S25B25+S26B25+S27B25+S28B25+S29B25+S30B25 - 6*B2311 <= 0;
S25B26+S26B26+S27B26+S28B26+S29B26+S30B26 - 6*B2312 <= 0;
S26B27+S29B27 - 2*B2321 <= 0;
S26B28+S29B28 - 2*B2322 <= 0;
S37B33+S38B33+S39B33+S40B33+S41B33+S42B33 - 6*B3111 <= 0;
S37B34+S38B34+S39B34+S40B34+S41B34+S42B34 - 6*B3112 <= 0;
S38B35+S41B35 - 2*B3121 <= 0;
S38B36+S41B36 - 2*B3122 <= 0;
S37S43B37+S37S44B37+S37S45B37+S38S43B37+S38S44B37+S38S45B37+S39S43B37+S39S44B37+S39S45B37+S40S43B37+
S40S44B37+S40S45B37+S41S43B37+S41S44B37+S41S45B37+S42S43B37+S42S44B37+S42S45B37 - 18*B3211 <= 0;
S37S43B38+S37S44B38+S37S45B38+S38S43B38+S38S44B38+S38S45B38+S39S43B38+S39S44B38+S39S45B38+S40S43B38+
S40S44B38+S40S45B38+S41S43B38+S41S44B38+S41S45B38+S42S43B38+S42S44B38+S42S45B38 - 18*B3212 <= 0;
S37S45B39+S38S45B39+S39S43B39+S39S44B39+S40S45B39+S41S45B39+S42S43B39+S42S44B39 - 8*B3221 <= 0;
S37S45B40+S38S45B40+S39S43B40+S39S44B40+S40S45B40+S41S45B40+S42S43B40+S42S44B40 - 8*B3222 <= 0;
S43S49B41+S43S50B41+S43S51B41+S44S49B41+S44S50B41+S44S51B41+S45S49B41+S45S50B41+S45S51B41+S43S52B41+
S43S53B41+S43S54B41+S44S52B41+S44S53B41+S44S54B41+S45S52B41+S45S53B41+S45S54B41 - 18*B3311 <= 0;
S43S49B42+S43S50B42+S43S51B42+S44S49B42+S44S50B42+S44S51B42+S45S49B42+S45S50B42+S45S51B42+S43S52B42+
S43S53B42+S43S54B42+S44S52B42+S44S53B42+S44S54B42+S45S52B42+S45S53B42+S45S54B42 - 18*B3312 <= 0;
S43S51B43+S44S51B43+S45S49B43+S45S50B43+S43S54B43+S44S54B43+S45S52B43+S45S53B43 - 8*B3321 <= 0;
S43S51B44+S44S51B44+S45S49B44+S45S50B44+S43S54B44+S44S54B44+S45S52B44+S45S53B44 - 8*B3322 <= 0;
S49B45+S50B45+S51B45+S52B45+S53B45+S54B45 - 6*B3411 <= 0;
S49B46+S50B46+S51B46+S52B46+S53B46+S54B46 - 6*B3412 <= 0;
S50B47+S53B47 - 2*B3421 <= 0;
S50B48+S53B48 - 2*B3422 <= 0;
S55B49+S56B49+S57B49+S58B49+S59B49+S60B49 - 6*B4111 <= 0;
S55B50+S56B50+S57B50+S58B50+S59B50+S60B50 - 6*B4112 <= 0;
S56B51+S59B51 - 2*B4121 <= 0;
S56B52+S59B52 - 2*B4122 <= 0;
S55S61B53+S55S62B53+S55S63B53+S56S61B53+S56S62B53+S56S63B53+S57S61B53+S57S62B53+S57S63B53+S58S61B53+
S58S62B53+S58S63B53+S59S61B53+S59S62B53+S59S63B53+S60S61B53+S60S62B53+S60S63B53 - 18*B4211 <= 0;
S55S61B54+S55S62B54+S55S63B54+S56S61B54+S56S62B54+S56S63B54+S57S61B54+S57S62B54+S57S63B54+S58S61B54+
S58S62B54+S58S63B54+S59S61B54+S59S62B54+S59S63B54+S60S61B54+S60S62B54+S60S63B54 - 18*B4212 <= 0;
S55S63B55+S56S63B55+S57S61B55+S57S62B55+S58S63B55+S59S63B55+S60S61B55+S60S62B55 - 8*B4221 <= 0;
S55S63B56+S56S63B56+S57S61B56+S57S62B56+S58S63B56+S59S63B56+S60S61B56+S60S62B56 - 8*B4222 <= 0;
S61B57+S62B57+S63B57 - 3*B4311 <= 0;
S61B58+S62B58+S63B58 - 3*B4312 <= 0;
S62B59 - 1*B4321 <= 0;
S62B60 - 1*B4322 <= 0;

!OPERATIONAL CONSTRAINTS;

!First Constraint - No Split Operations;
S1111+ S1112+ S1113+ S1121+ S1122+ S1123 = 1;
S1211+ S1212+ S1213 = 1;
S1311+ S1312+ S1313+ S1321+ S1322+ S1323 = 1;
S2111+ S2112+ S2113+ S2121+ S2122+ S2123 = 1;
S2211+ S2212+ S2213+ S2221+ S2222+ S2223 = 1;
S3111+ S3112+ S3113+ S3121+ S3122+ S3123 = 1;
S3211+ S3212+ S3213 = 1;
S3311+ S3312+ S3313+ S3321+ S3322+ S3323 = 1;
S4111+ S4112+ S4113+ S4121+ S4122+ S4123 = 1;
S4211+ S4212+ S4213 = 1;

!Second Constraint - Only One Unique Machine In Each Location;
X111+ X112+ X113 = 1;
X121+ X122+ X123 = 1;
X211+ X212+ X213 = 1;
X221+ X222+ X223 = 1;
X311+ X312+ X313 = 1;
X411+ X412+ X413 = 1;
X421+ X422+ X423 = 1;

```
X511+ X512+ X513 = 1;

!Third Constraint - Machine Capacity Limitations;
8*X111 - 4.0*S1111 - 7.0*S4111 >= 0;
8*X112 - 4.0*S1112 - 7.0*S4112 >= 0;
8*X113 - 4.0*S1113 - 7.0*S4113 >= 0;
8*X121 - 4.0*S1121 - 7.0*S4121 >= 0;
8*X122 - 4.0*S1122 - 7.0*S4122 >= 0;
8*X123 - 4.0*S1123 - 7.0*S4123 >= 0;
8*X211 - 3.5*S2111 - 4.0*S3111 - 6.0*S3311 >= 0;
8*X212 - 3.5*S2112 - 4.0*S3112 - 6.0*S3312 >= 0;
8*X213 - 3.5*S2113 - 4.0*S3113 - 6.0*S3313 >= 0;
8*X221 - 3.5*S2121 - 4.0*S3121 - 6.0*S3321 >= 0;
8*X222 - 3.5*S2122 - 4.0*S3122 - 6.0*S3322 >= 0;
8*X223 - 3.5*S2123 - 4.0*S3123 - 6.0*S3323 >= 0;
8*X311 - 5.0*S1211 - 3.0*S4211 >= 0;
8*X312 - 5.0*S1212 - 3.0*S4212 >= 0;
8*X313 - 5.0*S1213 - 3.0*S4213 >= 0;
8*X411 - 6.0*S1311 - 4.5*S2211 >= 0;
8*X412 - 6.0*S1312 - 4.5*S2212 >= 0;
8*X413 - 6.0*S1313 - 4.5*S2213 >= 0;
8*X421 - 6.0*S1321 - 4.5*S2221 >= 0;
8*X422 - 6.0*S1322 - 4.5*S2222 >= 0;
8*X423 - 6.0*S1323 - 4.5*S2223 >= 0;
8*X511 - 5.0*S3211 >= 0;
8*X512 - 5.0*S3212 >= 0;
8*X513 - 5.0*S3213 >= 0;

!Fourth Constraint - Maximum Number of Machines in a Location;
X111+ X121+ X211+ X221+ X311+ X411+ X421+ X511 <= 3;
X112+ X122+ X212+ X222+ X312+ X412+ X422+ X512 <= 3;
X113+ X123+ X213+ X223+ X313+ X413+ X423+ X513 <= 3;

!Fifth Constraint - Demand for each part;
!Part 1 - Operation 1;
S1B1+ 2*S1B2+ S2B1+ 2*S2B2+ S2B3+ 2*S2B4+ S3B1+ 2*S3B2+ S4B1+ 2*S4B2+ S5B1+ 2*S5B2+ S5B3+ 2*S5B4+ S6B1+
2*S6B2 = 3;
!Part 2 - Operation 1;
S19B17+ 2*S19B18+ S20B17+ 2*S20B18+ S20B19+ 2*S20B20+ S21B17+ 2*S21B18+ S22B17+ 2*S22B18+ S23B17+
2*S23B18+ S23B19+ 2*S23B20+ S24B17+ 2*S24B18 = 2;
!Part 3 - Operation 1;
S37B33+ 2*S37B34+ S38B33+ 2*S38B34+ S38B35+ 2*S38B36+ S39B33+ 2*S39B34+ S40B33+ 2*S40B34+ S41B33+
2*S41B34+ S41B35+ 2*S41B36+ S42B33+ 2*S42B34 = 3;
!Part 4 - Operation 1;
S55B49+ 2*S55B50+ S56B49+ 2*S56B50+ S56B51+ 2*S56B52+ S57B49+ 2*S57B50+ S58B49+ 2*S58B50+ S59B49+
2*S59B50+ S59B51+ 2*S59B52+ S60B49+ 2*S60B50 = 2;
!Part 1 - Operation 4;
S13B13+ 2*S13B14+ S14B13+ 2*S14B14+ S14B15+ 2*S14B16+ S15B13+ 2*S15B14+ S16B13+ 2*S16B14+ S17B13+
2*S17B14+ S17B15+ 2*S17B16+ S18B13+ 2*S18B14 = 3;
!Part 2 - Operation 3;
S25B25+ 2*S25B26+ S26B25+ 2*S26B26+ S26B27+ 2*S26B28+ S27B25+ 2*S27B26+ S28B25+ 2*S28B26+ S29B25+
2*S29B26+ S29B27+ 2*S29B28+ S30B25+ 2*S30B26 = 2;
!Part 3 - Operation 4;
S49B45+ 2*S49B46+ S50B45+ 2*S50B46+ S50B47+ 2*S50B48+ S51B45+ 2*S51B46+ S52B45+ 2*S52B46+ S53B45+
2*S53B46+ S53B47+ 2*S53B48+ S54B45+ 2*S54B46 = 3;
!Part 4 - Operation 3;
S61B57+ 2*S61B58+ S62B57+ 2*S62B58+ S62B59+ 2*S62B60+ S63B57+ 2*S63B58 = 2;
!Part 1 - Operation 2;
S1S7B5+ 2*S1S7B6+ S1S8B5+ 2*S1S8B6+ S1S9B5+ 2*S1S9B6+ S1S9B7+ 2*S1S9B8+ S2S7B5+ 2*S2S7B6+ S2S8B5+
2*S2S8B6+ S2S9B5+ 2*S2S9B6+ S2S9B7+ 2*S2S9B8+ S3S7B5+ 2*S3S7B6+ S3S7B7+ 2*S3S7B8+ S3S8B5+ 2*S3S8B6+
S3S8B7+ 2*S3S8B8+ S3S9B5+ 2*S3S9B6+ S4S7B5+ 2*S4S7B6+ S4S8B5+ 2*S4S8B6+ S4S9B5+ 2*S4S9B6+ S4S9B7+
2*S4S9B8+ S5S7B5+ 2*S5S7B6+ S5S8B5+ 2*S5S8B6+ S5S9B5+ 2*S5S9B6+ S5S9B7+ 2*S5S9B8+ S6S7B5+ 2*S6S7B6+
S6S7B7+ 2*S6S7B8+ S6S8B5+ 2*S6S8B6+ S6S8B7+ 2*S6S8B8+ S6S9B5+ 2*S6S9B6 = 3;
!Part 1 - Operation 3;
S7S13B9+ 2*S7S13B10+ S7S14B9+ 2*S7S14B10+ S7S15B9+ 2*S7S15B10+ S7S15B11+ 2*S7S15B12+ S8S13B9+ 2*S8S13B10+
S8S14B9+ 2*S8S14B10+ S8S15B9+ 2*S8S15B10+ S8S15B11+ 2*S8S15B12+ S9S13B9+ 2*S9S13B10+ S9S13B11+
2*S9S13B12+ S9S14B9+ 2*S9S14B10+ S9S14B11+ 2*S9S14B12+ S9S15B9+ 2*S9S15B10+ S7S16B9+ 2*S7S16B10+ S7S17B9+
2*S7S17B10+ S7S18B9+ 2*S7S18B10+ S7S18B11+ 2*S7S18B12+ S8S16B9+ 2*S8S16B10+ S8S17B9+ 2*S8S17B10+
S8S18B9+ 2*S8S18B10+ S8S18B11+ 2*S8S18B12+ S9S16B9+ 2*S9S16B10+ S9S16B11+ 2*S9S16B12+ S9S17B9+
2*S9S17B10+ S9S17B11+ 2*S9S17B12+ S9S18B9+ 2*S9S18B10 = 3;
```

!Part 2 - Operation 2;
S19S25B21+ 2*S19S25B22+ S19S26B21+ 2*S19S26B22+ S19S27B21+ 2*S19S27B22+ S19S27B23+ 2*S19S27B24+
S20S25B21+ 2*S20S25B22+ S20S26B21+ 2*S20S26B22+ S20S27B21+ 2*S20S27B22+ S20S27B23+ 2*S20S27B24+
S21S25B21+ 2*S21S25B22+ S21S25B23+ 2*S21S25B24+ S21S26B21+ 2*S21S26B22+ S21S26B23+ 2*S21S26B24+
S21S27B21+ 2*S21S27B22+ S19S28B21+ 2*S19S28B22+ S19S29B21+ 2*S19S29B22+ S19S30B21+ 2*S19S30B22+
S19S30B23+ 2*S19S30B24+ S20S28B21+ 2*S20S28B22+ S20S29B21+ 2*S20S29B22+ S20S30B21+ 2*S20S30B22+
S20S30B23+ 2*S20S30B24+ S21S28B21+ 2*S21S28B22+ S21S28B23+ 2*S21S28B24+ S21S29B21+ 2*S21S29B22+
S21S29B23+ 2*S21S29B24+ S21S30B21+ 2*S21S30B22+ S22S25B21+ 2*S22S25B22+ S22S26B21+ 2*S22S26B22+
S22S27B21+ 2*S22S27B22+ S22S27B23+ 2*S22S27B24+ S23S25B21+ 2*S23S25B22+ S23S26B21+ 2*S23S26B22+
S23S27B21+ 2*S23S27B22+ S23S27B23+ 2*S23S27B24+ S24S25B21+ 2*S24S25B22+ S24S25B23+ 2*S24S25B24+
S24S26B21+ 2*S24S26B22+ S24S26B23+ 2*S24S26B24+ S24S27B21+ 2*S24S27B22+ S22S28B21+ 2*S22S28B22+
S22S29B21+ 2*S22S29B22+ S22S30B21+ 2*S22S30B22+ S22S30B23+ 2*S22S30B24+ S23S28B21+ 2*S23S28B22+
S23S29B21+ 2*S23S29B22+ S23S30B21+ 2*S23S30B22+ S23S30B23+ 2*S23S30B24+ S24S28B21+ 2*S24S28B22+
S24S28B23+ 2*S24S28B24+ S24S29B21+ 2*S24S29B22+ S24S29B23+ 2*S24S29B24+ S24S30B21+ 2*S24S30B22 = 2;
!Part 3 - Operation 2;
S37S43B37+ 2*S37S43B38+ S37S44B37+ 2*S37S44B38+ S37S45B37+ 2*S37S45B38+ S37S45B39+ 2*S37S45B40+
S38S43B37+ 2*S38S43B38+ S38S44B37+ 2*S38S44B38+ S38S45B37+ 2*S38S45B38+ S38S45B39+ 2*S38S45B40+
S39S43B37+ 2*S39S43B38+ S39S43B39+ 2*S39S43B40+ S39S44B37+ 2*S39S44B38+ S39S44B39+ 2*S39S44B40+
S39S45B37+ 2*S39S45B38+ S40S43B37+ 2*S40S43B38+ S40S44B37+ 2*S40S44B38+ S40S45B37+ 2*S40S45B38+
S40S45B39+ 2*S40S45B40+ S41S43B37+ 2*S41S43B38+ S41S44B37+ 2*S41S44B38+ S41S45B37+ 2*S41S45B38+
S41S45B39+ 2*S41S45B40+ S42S43B37+ 2*S42S43B38+ S42S43B39+ 2*S42S43B40+ S42S44B37+ 2*S42S44B38+
S42S44B39+ 2*S42S44B40+ S42S45B37+ 2*S42S45B38 = 3;
!Part 3 - Operation 3;
S43S49B41+ 2*S43S49B42+ S43S50B41+ 2*S43S50B42+ S43S51B41+ 2*S43S51B42+ S43S51B43+ 2*S43S51B44+
S44S49B41+ 2*S44S49B42+ S44S50B41+ 2*S44S50B42+ S44S51B41+ 2*S44S51B42+ S44S51B43+ 2*S44S51B44+
S45S49B41+ 2*S45S49B42+ S45S49B43+ 2*S45S49B44+ S45S50B41+ 2*S45S50B42+ S45S50B43+ 2*S45S50B44+
S45S51B41+ 2*S45S51B42+ S43S52B41+ 2*S43S52B42+ S43S53B41+ 2*S43S53B42+ S43S54B41+ 2*S43S54B42+
S43S54B43+ 2*S43S54B44+ S44S52B41+ 2*S44S52B42+ S44S53B41+ 2*S44S53B42+ S44S54B41+ 2*S44S54B42+
S44S54B43+ 2*S44S54B44+ S45S52B41+ 2*S45S52B42+ S45S52B43+ 2*S45S52B44+ S45S53B41+ 2*S45S53B42+
S45S53B43+ 2*S45S53B44+ S45S54B41+ 2*S45S54B42 = 3;
!Part 4 - Operation 2;
S55S61B53+ 2*S55S61B54+ S55S62B53+ 2*S55S62B54+ S55S63B53+ 2*S55S63B54+ S55S63B55+ 2*S55S63B56+
S56S61B53+ 2*S56S61B54+ S56S62B53+ 2*S56S62B54+ S56S63B53+ 2*S56S63B54+ S56S63B55+ 2*S56S63B56+
S57S61B53+ 2*S57S61B54+ S57S61B55+ 2*S57S61B56+ S57S62B53+ 2*S57S62B54+ S57S62B55+ 2*S57S62B56+
S57S63B53+ 2*S57S63B54+ S58S61B53+ 2*S58S61B54+ S58S62B53+ 2*S58S62B54+ S58S63B53+ 2*S58S63B54+
S58S63B55+ 2*S58S63B56+ S59S61B53+ 2*S59S61B54+ S59S62B53+ 2*S59S62B54+ S59S63B53+ 2*S59S63B54+
S59S63B55+ 2*S59S63B56+ S60S61B53+ 2*S60S61B54+ S60S61B55+ 2*S60S61B56+ S60S62B53+ 2*S60S62B54+
S60S62B55+ 2*S60S62B56+ S60S63B53+ 2*S60S63B54 = 2;

!Sixth Constraint - Demand for each AGV;
!Constraint for AGV 1;
48*S1B1+ 96*S1B2+ 48*S2B1+ 96*S2B2+ 8*S2B3+ 16*S2B4+ 8*S3B1+ 16*S3B2+ 48*S4B1+ 96*S4B2+ 48*S5B1+ 96*S5B2+
8*S5B3+ 16*S5B4+ 8*S6B1+ 16*S6B2+ 48*S19B17+ 96*S19B18+ 48*S20B17+ 96*S20B18+ 8*S20B19+ 16*S20B20+
8*S21B17+ 16*S21B18+ 48*S22B17+ 96*S22B18+ 48*S23B17+ 96*S23B18+ 8*S23B19+ 16*S23B20+ 8*S24B17+ 16*S24B18+
48*S37B33+ 96*S37B34+ 48*S38B33+ 96*S38B34+ 8*S38B35+ 16*S38B36+ 8*S39B33+ 16*S39B34+ 48*S40B33+
96*S40B34+ 48*S41B33+ 96*S41B34+ 8*S41B35+ 16*S41B36+ 8*S42B33+ 16*S42B34+ 48*S55B49+ 96*S55B50+
48*S56B49+ 96*S56B50+ 8*S56B51+ 16*S56B52+ 8*S57B49+ 16*S57B50+ 48*S58B49+ 96*S58B50+ 48*S59B49+
96*S59B50+ 8*S59B51+ 16*S59B52+ 8*S60B49+ 16*S60B50+ 48*S13B13+ 96*S13B14+ 8*S14B13+ 16*S14B14+ 48*S14B15+
96*S14B16+ 8*S15B13+ 16*S15B14+ 48*S16B13+ 96*S16B14+ 8*S17B13+ 16*S17B14+ 48*S17B15+ 96*S17B16+ 8*S18B13+
16*S18B14+ 48*S25B25+ 96*S25B26+ 8*S26B25+ 16*S26B26+ 48*S26B27+ 96*S26B28+ 8*S27B25+ 16*S27B26+
48*S28B25+ 96*S28B26+ 8*S29B25+ 16*S29B26+ 48*S29B27+ 96*S29B28+ 8*S30B25+ 16*S30B26+ 48*S49B45+
96*S49B46+ 8*S50B45+ 16*S50B46+ 48*S50B47+ 96*S50B48+ 8*S51B45+ 16*S51B46+ 48*S52B45+ 96*S52B46+ 8*S53B45+
16*S53B46+ 48*S53B47+ 96*S53B48+ 8*S54B45+ 16*S54B46+ 48*S61B57+ 96*S61B58+ 8*S62B57+ 16*S62B58+
48*S62B59+ 96*S62B60+ 8*S63B57+ 16*S63B58+ 41*S1S8B5+ 82*S1S8B6+ 51*S1S9B5+ 102*S1S9B6+ 41*S1S9B7+
82*S1S9B8+ 41*S2S7B5+ 82*S2S7B6+ 46*S2S9B7+ 92*S2S9B8+ 51*S3S7B5+ 102*S3S7B6+ 41*S3S7B7+ 82*S3S7B8+
46*S3S8B7+ 92*S3S8B8+ 41*S4S8B5+ 82*S4S8B6+ 51*S4S9B5+ 102*S4S9B6+ 41*S4S9B7+ 82*S4S9B8+ 41*S5S7B5+
82*S5S7B6+ 46*S5S9B7+ 92*S5S9B8+ 51*S6S7B5+ 102*S6S7B6+ 41*S6S7B7+ 82*S6S7B8+ 46*S6S8B7+ 92*S6S8B8+
41*S7S14B9+ 82*S7S14B10+ 51*S7S15B9+ 102*S7S15B10+ 41*S7S15B11+ 82*S7S15B12+ 41*S8S13B9+ 82*S8S13B10+
46*S8S15B11+ 92*S8S15B12+ 51*S9S13B9+ 102*S9S13B10+ 41*S9S13B11+ 82*S9S13B12+ 46*S9S14B11+ 92*S9S14B12+
41*S7S17B9+ 82*S7S17B10+ 51*S7S18B9+ 102*S7S18B10+ 41*S7S18B11+ 82*S7S18B12+ 41*S8S16B9+ 82*S8S16B10+
46*S8S18B11+ 92*S8S18B12+ 51*S9S16B9+ 102*S9S16B10+ 41*S9S16B11+ 82*S9S16B12+ 46*S9S17B11+ 92*S9S17B12+
41*S19S26B21+ 82*S19S26B22+ 51*S19S27B21+ 102*S19S27B22+ 41*S19S27B23+82*S19S27B24+ 41*S20S25B21+
82*S20S25B22+ 46*S20S27B23+ 92*S20S27B24+ 51*S21S25B21+ 102*S21S25B22+ 41*S21S25B23+ 82*S21S25B24+
46*S21S26B23+ 92*S21S26B24+ 41*S19S29B21+ 82*S19S29B22+ 51*S19S30B21+ 102*S19S30B22+ 41*S19S30B23+
82*S19S30B24+ 41*S20S28B21+ 82*S20S28B22+ 46*S20S30B23+ 92*S20S30B24+ 51*S21S28B21+ 102*S21S28B22+
41*S21S28B23+ 82*S21S28B24+ 46*S21S29B23+ 92*S21S29B24+ 41*S22S26B21+ 82*S22S26B22+ 51*S22S27B21+
102*S22S27B22+ 41*S22S27B23+ 82*S22S27B24+ 41*S23S25B21+ 82*S23S25B22+ 46*S23S27B23+ 92*S23S27B24+
51*S24S25B21+ 102*S24S25B22+ 41*S24S25B23+ 82*S24S25B24+ 46*S24S26B23+ 92*S24S26B24+ 41*S22S29B21+
82*S22S29B22+ 51*S22S30B21+ 102*S22S30B22+ 41*S22S30B23+ 82*S22S30B24+ 41*S23S28B21+ 82*S23S28B22+
46*S23S30B23+ 92*S23S30B24+ 51*S24S28B21+ 102*S24S28B22+ 41*S24S28B23+ 82*S24S28B24+ 46*S24S29B23+

92*S24S29B24+ 41*S37S44B37+ 82*S37S44B38+ 51*S37S45B37+ 102*S37S45B38+ 41*S37S45B39+ 82*S37S45B40+
41*S38S43B37+ 82*S38S43B38+ 46*S38S45B39+ 92*S38S45B40+ 51*S39S43B37+ 102*S39S43B38+ 41*S39S43B39+
82*S39S43B40+ 46*S39S44B39+ 92*S39S44B40+ 41*S40S44B37+ 82*S40S44B38+ 51*S40S45B37+ 102*S40S45B38+
41*S40S45B39+ 82*S40S45B40+ 41*S41S43B37+ 82*S41S43B38+ 46*S41S45B39+ 92*S41S45B40+ 51*S42S43B37+
102*S42S43B38+ 41*S42S43B39+ 82*S42S43B40+ 46*S42S44B39+ 92*S42S44B40+ 41*S43S50B41+ 82*S43S50B42+
51*S43S51B41+ 102*S43S51B42+ 41*S43S51B43+ 82*S43S51B44+ 41*S44S49B41+ 82*S44S49B42+ 46*S44S51B43+
92*S44S51B44+ 51*S45S49B41+ 102*S45S49B42+ 41*S45S49B43+ 82*S45S49B44+ 46*S45S50B43+ 92*S45S50B44+
41*S43S53B41+ 82*S43S53B42+ 51*S43S54B41+ 102*S43S54B42+ 41*S43S54B43+ 82*S43S54B44+ 41*S44S52B41+
82*S44S52B42+ 46*S44S54B43+ 92*S44S54B44+ 51*S45S52B41+ 102*S45S52B42+ 41*S45S52B43+ 82*S45S52B44+
46*S45S53B43+ 92*S45S53B44+ 41*S55S62B53+ 82*S55S62B54+ 51*S55S63B53+ 102*S55S63B54+ 41*S55S63B55+
82*S55S63B56+ 41*S56S61B53+ 82*S56S61B54+ 46*S56S63B55+ 92*S56S63B56+ 51*S57S61B53+ 102*S57S61B54+
41*S57S61B55+ 82*S57S61B56+ 46*S57S62B55+ 92*S57S62B56+ 41*S58S62B53+ 82*S58S62B54+ 51*S58S63B53+
102*S58S63B54+ 41*S58S63B55+ 82*S58S63B56+ 41*S59S61B53+ 82*S59S61B54+ 46*S59S63B55+ 92*S59S63B56+
51*S60S61B53+ 102*S60S61B54+ 41*S60S61B55+ 82*S60S61B56+ 46*S60S62B55+ 92*S60S62B56 <= 10000;

!Constraint for AGV 2;
16*S2B1+ 32*S2B2+ 32*S2B3+ 64*S2B4+ 16*S5B1+ 32*S5B2+ 32*S5B3+ 64*S5B4+ 16*S20B17+ 32*S20B18+ 32*S20B19+
64*S20B20+ 16*S23B17+ 32*S23B18+ 32*S23B19+ 64*S23B20+ 16*S38B33+ 32*S38B34+ 32*S38B35+ 64*S38B36+
16*S41B33+ 32*S41B34+ 32*S41B35+ 64*S41B36+ 16*S56B49+ 32*S56B50+ 32*S56B51+ 64*S56B52+ 16*S59B49+
32*S59B50+ 32*S59B51+ 64*S59B52+ 32*S14B13+ 64*S14B14+ 16*S14B15+ 32*S14B16+ 32*S17B13+ 64*S17B14+
16*S17B15+ 32*S17B16+ 32*S26B25+ 64*S26B26+ 16*S26B27+ 32*S26B28+ 32*S29B25+ 64*S29B26+ 16*S29B27+
32*S29B28+ 32*S50B45+ 64*S50B46+ 16*S50B47+ 32*S50B48+ 32*S53B45+ 64*S53B46+ 16*S53B47+ 32*S53B48+
32*S62B57+ 64*S62B58+ 16*S62B59+ 32*S62B60+ 16*S1S8B5+ 32*S1S8B6+ 33*S1S9B7+ 66*S1S9B8+ 16*S2S7B5+
32*S2S7B6+ 32*S2S9B5+ 64*S2S9B6+ 16*S2S9B7+ 32*S2S9B8+ 33*S3S7B7+ 66*S3S7B8+ 32*S3S8B5+ 64*S3S8B6+
16*S3S8B7+ 32*S3S8B8+ 16*S4S8B5+ 32*S4S8B6+ 33*S4S9B7+ 66*S4S9B8+ 16*S5S7B5+ 32*S5S7B6+ 32*S5S9B5+
64*S5S9B6+ 16*S5S9B7+ 32*S5S9B8+ 33*S6S7B7+ 66*S6S7B8+ 32*S6S8B5+ 64*S6S8B6+ 16*S6S8B7+ 32*S6S8B8+
16*S7S14B9+ 32*S7S14B10+ 33*S7S15B11+ 66*S7S15B12+ 16*S8S13B9+ 32*S8S13B10+ 32*S8S15B9+ 64*S8S15B10+
16*S8S15B11+ 32*S8S15B12+ 33*S9S13B11+ 66*S9S13B12+ 32*S9S14B9+ 64*S9S14B10+ 16*S9S14B11+ 32*S9S14B12+
16*S7S17B9+ 32*S7S17B10+ 33*S7S18B11+ 66*S7S18B12+ 16*S8S16B9+ 32*S8S16B10+ 32*S8S18B9+ 64*S8S18B10+
16*S8S18B11+ 32*S8S18B12+ 33*S9S16B11+ 66*S9S16B12+ 32*S9S17B9+ 64*S9S17B10+ 16*S9S17B11+ 32*S9S17B12+
16*S19S26B21+ 32*S19S26B22+ 33*S19S27B23+ 66*S19S27B24+ 16*S20S25B21+ 32*S20S25B22+ 32*S20S27B21+
64*S20S27B22+ 16*S20S27B23+ 32*S20S27B24+ 33*S21S25B23+ 66*S21S25B24+ 32*S21S26B21+ 64*S21S26B22+
16*S21S26B23+ 32*S21S26B24+ 16*S19S29B21+ 32*S19S29B22+ 33*S19S30B23+ 66*S19S30B24+ 16*S20S28B21+
32*S20S28B22+ 32*S20S30B21+ 64*S20S30B22+ 16*S20S30B23+ 32*S20S30B24+ 33*S21S28B23+ 66*S21S28B24+
32*S21S29B21+ 64*S21S29B22+ 16*S21S29B23+ 32*S21S29B24+ 16*S22S26B21+ 32*S22S26B22+ 33*S22S27B23+
66*S22S27B24+ 16*S23S25B21+ 32*S23S25B22+ 32*S23S27B21+ 64*S23S27B22+ 16*S23S27B23+ 32*S23S27B24+
33*S24S25B23+ 66*S24S25B24+ 32*S24S26B21+ 64*S24S26B22+ 16*S24S26B23+ 32*S24S26B24+ 16*S22S29B21+
32*S22S29B22+ 33*S22S30B23+ 66*S22S30B24+ 16*S23S28B21+ 32*S23S28B22+ 32*S23S30B21+ 64*S23S30B22+
16*S23S30B23+ 32*S23S30B24+ 33*S24S28B23+ 66*S24S28B24+ 32*S24S29B21+ 64*S24S29B22+ 16*S24S29B23+
32*S24S29B24+ 16*S37S44B37+ 32*S37S44B38+ 33*S37S45B39+ 66*S37S45B40+ 16*S38S43B37+ 32*S38S43B38+
32*S38S45B37+ 64*S38S45B38+ 16*S38S45B39+ 32*S38S45B40+ 33*S39S43B39+ 66*S39S43B40+ 32*S39S44B37+
64*S39S44B38+ 16*S39S44B39+ 32*S39S44B40+ 16*S40S44B37+ 32*S40S44B38+ 33*S40S45B39+ 66*S40S45B40+
16*S41S43B37+ 32*S41S43B38+ 32*S41S45B37+ 64*S41S45B38+ 16*S41S45B39+ 32*S41S45B40+ 33*S42S43B39+
66*S42S43B40+ 32*S42S44B37+ 64*S42S44B38+ 16*S42S44B39+ 32*S42S44B40+ 16*S43S50B41+ 32*S43S50B42+
33*S43S51B43+ 66*S43S51B44+ 16*S44S49B41+ 32*S44S49B42+ 32*S44S51B41+ 64*S44S51B42+ 16*S44S51B43+
32*S44S51B44+ 33*S45S49B43+ 66*S45S49B44+ 32*S45S50B41+ 64*S45S50B42+ 16*S45S50B43+ 32*S45S50B44+
16*S43S53B41+ 32*S43S53B42+ 33*S43S54B43+ 66*S43S54B44+ 16*S44S52B41+ 32*S44S52B42+ 32*S44S54B41+
64*S44S54B42+ 16*S44S54B43+ 32*S44S54B44+ 33*S45S52B43+ 66*S45S52B44+ 32*S45S53B41+ 64*S45S53B42+
16*S45S53B43+ 32*S45S53B44+ 16*S55S62B53+ 32*S55S62B54+ 33*S55S63B55+ 66*S55S63B56+ 16*S56S61B53+
32*S56S61B54+ 32*S56S63B53+ 64*S56S63B54+ 16*S56S63B55+ 32*S56S63B56+ 33*S57S61B55+ 66*S57S61B56+
32*S57S62B53+ 64*S57S62B54+ 16*S57S62B55+ 32*S57S62B56+ 16*S58S62B53+ 32*S58S62B54+ 33*S58S63B55+
66*S58S63B56+ 16*S59S61B53+ 32*S59S61B54+ 32*S59S63B53+ 64*S59S63B54+ 16*S59S63B55+ 32*S59S63B56+
33*S60S61B55+ 66*S60S61B56+ 32*S60S62B53+ 64*S60S62B54+ 16*S60S62B55+ 32*S60S62B56 <= 10000;

!Constraint for AGV 3;
33*S2B3+ 66*S2B4+ 16*S3B1+ 32*S3B2+ 33*S5B3+ 66*S5B4+ 16*S6B1+ 32*S6B2+ 33*S20B19+ 66*S20B20+ 16*S21B17+
32*S21B18+ 33*S23B19+ 66*S23B20+ 16*S24B17+ 32*S24B18+ 33*S38B35+ 66*S38B36+ 16*S39B33+ 32*S39B34+
33*S41B35+ 66*S41B36+ 16*S42B33+ 32*S42B34+ 33*S56B51+ 66*S56B52+ 16*S57B49+ 32*S57B50+ 33*S59B51+
66*S59B52+ 16*S60B49+ 32*S60B50+ 33*S14B13+ 66*S14B14+ 16*S15B13+ 32*S15B14+ 33*S17B13+ 66*S17B14+
16*S18B13+ 32*S18B14+ 33*S26B25+ 66*S26B26+ 16*S27B25+ 32*S27B26+ 33*S29B25+ 66*S29B26+ 16*S30B25+
32*S30B26+ 33*S50B45+ 66*S50B46+ 16*S51B45+ 32*S51B46+ 33*S53B45+ 66*S53B46+ 16*S54B45+ 32*S54B46+
33*S62B57+ 66*S62B58+ 16*S63B57+ 32*S63B58+ 16*S1S9B5+ 32*S1S9B6+ 32*S1S9B7+ 64*S1S9B8+ 32*S2S9B5+
64*S2S9B6+ 16*S2S9B7+ 32*S2S9B8+ 16*S3S7B5+ 32*S3S7B6+ 32*S3S7B7+ 64*S3S7B8+ 32*S3S8B5+ 64*S3S8B6+
16*S3S8B7+ 32*S3S8B8+ 16*S4S9B5+ 32*S4S9B6+ 32*S4S9B7+ 64*S4S9B8+ 32*S5S9B5+ 64*S5S9B6+ 16*S5S9B7+
32*S5S9B8+ 16*S6S7B5+ 32*S6S7B6+ 32*S6S7B7+ 64*S6S7B8+ 32*S6S8B5+ 64*S6S8B6+ 16*S6S8B7+ 32*S6S8B8+
16*S7S15B9+ 32*S7S15B10+ 32*S7S15B11+ 64*S7S15B12+ 32*S8S15B9+ 64*S8S15B10+ 16*S8S15B11+ 32*S8S15B12+
16*S9S13B9+ 32*S9S13B10+ 32*S9S13B11+ 64*S9S13B12+ 32*S9S14B9+ 64*S9S14B10+ 16*S9S14B11+ 32*S9S14B12+
16*S7S18B9+ 32*S7S18B10+ 32*S7S18B11+ 64*S7S18B12+ 32*S8S18B9+ 64*S8S18B10+ 16*S8S18B11+ 32*S8S18B12+
16*S9S16B9+ 32*S9S16B10+ 32*S9S16B11+ 64*S9S16B12+ 32*S9S17B9+ 64*S9S17B10+ 16*S9S17B11+ 32*S9S17B12+
16*S19S27B21+ 32*S19S27B22+ 32*S19S27B23+ 64*S19S27B24+ 32*S20S27B21+ 64*S20S27B22+ 16*S20S27B23+

```
32*S20S27B24+ 16*S21S25B21+ 32*S21S25B22+ 32*S21S25B23+ 64*S21S25B24+ 32*S21S26B21+ 64*S21S26B22+
16*S21S26B23+ 32*S21S26B24+ 16*S19S30B21+ 32*S19S30B22+ 32*S19S30B23+ 64*S19S30B24+ 32*S20S30B21+
64*S20S30B22+ 16*S20S30B23+ 32*S20S30B24+ 16*S21S28B21+ 32*S21S28B22+ 32*S21S28B23+ 64*S21S28B24+
32*S21S29B21+ 64*S21S29B22+ 16*S21S29B23+ 32*S21S29B24+ 16*S22S27B21+ 32*S22S27B22+ 32*S22S27B23+
64*S22S27B24+ 32*S23S27B21+ 64*S23S27B22+ 16*S23S27B23+ 32*S23S27B24+ 16*S24S25B21+ 32*S24S25B22+
32*S24S25B23+ 64*S24S25B24+ 32*S24S26B21+ 64*S24S26B22+ 16*S24S26B23+ 32*S24S26B24+ 16*S22S30B21+
32*S22S30B22+ 32*S22S30B23+ 64*S22S30B24+ 32*S23S30B21+ 64*S23S30B22+ 16*S23S30B23+ 32*S23S30B24+
16*S24S28B21+ 32*S24S28B22+ 32*S24S28B23+ 64*S24S28B24+ 32*S24S29B21+ 64*S24S29B22+ 16*S24S29B23+
32*S24S29B24+ 16*S37S45B37+ 32*S37S45B38+ 32*S37S45B39+ 64*S37S45B40+ 32*S38S45B37+ 64*S38S45B38+
16*S38S45B39+ 32*S38S45B40+ 16*S39S43B37+ 32*S39S43B38+ 32*S39S43B39+ 64*S39S43B40+ 32*S39S44B37+
64*S39S44B38+ 16*S39S44B39+ 32*S39S44B40+ 16*S40S45B37+ 32*S40S45B38+ 32*S40S45B39+ 64*S40S45B40+
32*S41S45B37+ 64*S41S45B38+ 16*S41S45B39+ 32*S41S45B40+ 16*S42S43B37+ 32*S42S43B38+ 32*S42S43B39+
64*S42S43B40+ 32*S42S44B37+ 64*S42S44B38+ 16*S42S44B39+ 32*S42S44B40+ 16*S43S51B41+ 32*S43S51B42+
32*S43S51B43+ 64*S43S51B44+ 32*S44S51B41+ 64*S44S51B42+ 16*S44S51B43+ 32*S44S51B44+ 16*S45S49B41+
32*S45S49B42+ 32*S45S49B43+ 64*S45S49B44+ 32*S45S50B41+ 64*S45S50B42+ 16*S45S50B43+ 32*S45S50B44+
16*S43S54B41+ 32*S43S54B42+ 32*S43S54B43+ 64*S43S54B44+ 32*S44S54B41+ 64*S44S54B42+ 16*S44S54B43+
32*S44S54B44+ 16*S45S52B41+ 32*S45S52B42+ 32*S45S52B43+ 64*S45S52B44+ 32*S45S53B41+ 64*S45S53B42+
16*S45S53B43+ 32*S45S53B44+ 16*S55S63B53+ 32*S55S63B54+ 32*S55S63B55+ 64*S55S63B56+ 32*S56S63B53+
64*S56S63B54+ 16*S56S63B55+ 32*S56S63B56+ 16*S57S61B53+ 32*S57S61B54+ 32*S57S61B55+ 64*S57S61B56+
32*S57S62B53+ 64*S57S62B54+ 16*S57S62B55+ 32*S57S62B56+ 16*S58S63B53+ 32*S58S63B54+ 32*S58S63B55+
64*S58S63B56+ 32*S59S63B53+ 64*S59S63B54+ 16*S59S63B55+ 32*S59S63B56+ 16*S60S61B53+ 32*S60S61B54+
32*S60S61B55+ 64*S60S61B56+ 32*S60S62B53+ 64*S60S62B54+ 16*S60S62B55+ 32*S60S62B56 <= 10000;

!Declare the SB Variable as Real Variables bounded between 0 and 1;  .
@BND(0,S1B1,1);   @BND(0,S1B2,1);   @BND(0,S2B1,1);   @BND(0,S2B2,1);   @BND(0,S2B3,1);
@BND(0,S2B4,1);   @BND(0,S3B1,1);   @BND(0,S3B2,1);   @BND(0,S4B1,1);   @BND(0,S4B2,1);
@BND(0,S5B1,1);   @BND(0,S5B2,1);   @BND(0,S5B3,1);   @BND(0,S5B4,1);   @BND(0,S6B1,1);
@BND(0,S6B2,1);   @BND(0,S19B17,1);  @BND(0,S19B18,1);  @BND(0,S20B17,1);  @BND(0,S20B18,1);
@BND(0,S20B19,1);  @BND(0,S20B20,1);  @BND(0,S21B17,1);  @BND(0,S21B18,1);  @BND(0,S22B17,1);
@BND(0,S22B18,1);  @BND(0,S23B17,1);  @BND(0,S23B18,1);  @BND(0,S23B19,1);  @BND(0,S23B20,1);
@BND(0,S24B17,1);  @BND(0,S24B18,1);  @BND(0,S37B33,1);  @BND(0,S37B34,1);  @BND(0,S38B33,1);
@BND(0,S38B34,1);  @BND(0,S38B35,1);  @BND(0,S38B36,1);  @BND(0,S39B33,1);  @BND(0,S39B34,1);
@BND(0,S40B33,1);  @BND(0,S40B34,1);  @BND(0,S41B33,1);  @BND(0,S41B34,1);  @BND(0,S41B35,1);
@BND(0,S41B36,1);  @BND(0,S42B33,1);  @BND(0,S42B34,1);  @BND(0,S55B49,1);  @BND(0,S55B50,1);
@BND(0,S56B49,1);  @BND(0,S56B50,1);  @BND(0,S56B51,1);  @BND(0,S56B52,1);  @BND(0,S57B49,1);
@BND(0,S57B50,1);  @BND(0,S58B49,1);  @BND(0,S58B50,1);  @BND(0,S59B49,1);  @BND(0,S59B50,1);
@BND(0,S59B51,1);  @BND(0,S59B52,1);  @BND(0,S60B49,1);  @BND(0,S60B50,1);  @BND(0,S13B13,1);
@BND(0,S13B14,1);  @BND(0,S14B13,1);  @BND(0,S14B14,1);  @BND(0,S14B15,1);  @BND(0,S14B16,1);
@BND(0,S15B13,1);  @BND(0,S15B14,1);  @BND(0,S16B13,1);  @BND(0,S16B14,1);  @BND(0,S17B13,1);
@BND(0,S17B14,1);  @BND(0,S17B15,1);  @BND(0,S17B16,1);  @BND(0,S18B13,1);  @BND(0,S18B14,1);
@BND(0,S25B25,1);  @BND(0,S25B26,1);  @BND(0,S26B25,1);  @BND(0,S26B26,1);  @BND(0,S26B27,1);
@BND(0,S26B28,1);  @BND(0,S27B25,1);  @BND(0,S27B26,1);  @BND(0,S28B25,1);  @BND(0,S28B26,1);
@BND(0,S29B25,1);  @BND(0,S29B26,1);  @BND(0,S29B27,1);  @BND(0,S29B28,1);  @BND(0,S30B25,1);
@BND(0,S30B26,1);  @BND(0,S49B45,1);  @BND(0,S49B46,1);  @BND(0,S50B45,1);  @BND(0,S50B46,1);
@BND(0,S50B47,1);  @BND(0,S50B48,1);  @BND(0,S51B45,1);  @BND(0,S51B46,1);  @BND(0,S52B45,1);
@BND(0,S52B46,1);  @BND(0,S53B45,1);  @BND(0,S53B46,1);  @BND(0,S53B47,1);  @BND(0,S53B48,1);
@BND(0,S54B45,1);  @BND(0,S54B46,1);  @BND(0,S61B57,1);  @BND(0,S61B58,1);  @BND(0,S62B57,1);
@BND(0,S62B58,1);  @BND(0,S62B59,1);  @BND(0,S62B60,1);  @BND(0,S63B57,1);  @BND(0,S63B58,1);
@BND(0,S1S7B5,1);  @BND(0,S1S7B6,1);  @BND(0,S1S8B5,1);  @BND(0,S1S8B6,1);  @BND(0,S1S9B5,1);
@BND(0,S1S9B6,1);  @BND(0,S1S9B7,1);  @BND(0,S1S9B8,1);  @BND(0,S2S7B5,1);  @BND(0,S2S7B6,1);
@BND(0,S2S8B5,1);  @BND(0,S2S8B6,1);  @BND(0,S2S9B5,1);  @BND(0,S2S9B6,1);  @BND(0,S2S9B7,1);
@BND(0,S2S9B8,1);  @BND(0,S3S7B5,1);  @BND(0,S3S7B6,1);  @BND(0,S3S7B7,1);  @BND(0,S3S7B8,1);
@BND(0,S3S8B5,1);  @BND(0,S3S8B6,1);  @BND(0,S3S8B7,1);  @BND(0,S3S8B8,1);  @BND(0,S3S9B5,1);
@BND(0,S3S9B6,1);  @BND(0,S4S7B5,1);  @BND(0,S4S7B6,1);  @BND(0,S4S8B5,1);  @BND(0,S4S8B6,1);
@BND(0,S4S9B5,1);  @BND(0,S4S9B6,1);  @BND(0,S4S9B7,1);  @BND(0,S4S9B8,1);  @BND(0,S5S7B5,1);
@BND(0,S5S7B6,1);  @BND(0,S5S8B5,1);  @BND(0,S5S8B6,1);  @BND(0,S5S9B5,1);  @BND(0,S5S9B6,1);
@BND(0,S5S9B7,1);  @BND(0,S5S9B8,1);  @BND(0,S6S7B5,1);  @BND(0,S6S7B6,1);  @BND(0,S6S7B7,1);
@BND(0,S6S7B8,1);  @BND(0,S6S8B5,1);  @BND(0,S6S8B6,1);  @BND(0,S6S8B7,1);  @BND(0,S6S8B8,1);
@BND(0,S6S9B5,1);  @BND(0,S6S9B6,1);  @BND(0,S7S13B9,1);  @BND(0,S7S13B10,1);
@BND(0,S7S14B9,1);  @BND(0,S7S14B10,1);  @BND(0,S7S15B9,1);  @BND(0,S7S15B10,1);
@BND(0,S7S15B11,1);  @BND(0,S7S15B12,1);  @BND(0,S8S13B9,1);  @BND(0,S8S13B10,1);
@BND(0,S8S14B9,1);  @BND(0,S8S14B10,1);  @BND(0,S8S15B9,1);  @BND(0,S8S15B10,1);
@BND(0,S8S15B11,1);  @BND(0,S8S15B12,1);  @BND(0,S9S13B9,1);  @BND(0,S9S13B10,1);
@BND(0,S9S13B11,1);  @BND(0,S9S13B12,1);  @BND(0,S9S14B9,1);  @BND(0,S9S14B10,1);
@BND(0,S9S14B11,1);  @BND(0,S9S14B12,1);  @BND(0,S9S15B9,1);  @BND(0,S9S15B10,1);
@BND(0,S7S16B9,1);  @BND(0,S7S16B10,1);  @BND(0,S7S17B9,1);  @BND(0,S7S17B10,1);
@BND(0,S7S18B9,1);  @BND(0,S7S18B10,1);  @BND(0,S7S18B11,1);  @BND(0,S7S18B12,1);
@BND(0,S8S16B9,1);  @BND(0,S8S16B10,1);  @BND(0,S8S17B9,1);  @BND(0,S8S17B10,1);
@BND(0,S8S18B9,1);  @BND(0,S8S18B10,1);  @BND(0,S8S18B11,1);  @BND(0,S8S18B12,1);
@BND(0,S9S16B9,1);  @BND(0,S9S16B10,1);  @BND(0,S9S16B11,1);  @BND(0,S9S16B12,1);
```

```
@BND(0,S9S17B9,1);    @BND(0,S9S17B10,1);   @BND(0,S9S17B11,1);   @BND(0,S9S17B12,1);
@BND(0,S9S18B9,1);    @BND(0,S9S18B10,1);   @BND(0,S19S25B21,1);  @BND(0,S19S25B22,1);
@BND(0,S19S26B21,1);  @BND(0,S19S26B22,1);  @BND(0,S19S27B21,1);  @BND(0,S19S27B22,1);
@BND(0,S19S27B23,1);  @BND(0,S19S27B24,1);  @BND(0,S20S25B21,1);  @BND(0,S20S25B22,1);
@BND(0,S20S26B21,1);  @BND(0,S20S26B22,1);  @BND(0,S20S27B21,1);  @BND(0,S20S27B22,1);
@BND(0,S20S27B23,1);  @BND(0,S20S27B24,1);  @BND(0,S21S25B21,1);  @BND(0,S21S25B22,1);
@BND(0,S21S25B23,1);  @BND(0,S21S25B24,1);  @BND(0,S21S26B21,1);  @BND(0,S21S26B22,1);
@BND(0,S21S26B23,1);  @BND(0,S21S26B24,1);  @BND(0,S21S27B21,1);  @BND(0,S21S27B22,1);
@BND(0,S19S28B21,1);  @BND(0,S19S28B22,1);  @BND(0,S19S29B21,1);  @BND(0,S19S29B22,1);
@BND(0,S19S30B21,1);  @BND(0,S19S30B22,1);  @BND(0,S19S30B23,1);  @BND(0,S19S30B24,1);
@BND(0,S20S28B21,1);  @BND(0,S20S28B22,1);  @BND(0,S20S29B21,1);  @BND(0,S20S29B22,1);
@BND(0,S20S30B21,1);  @BND(0,S20S30B22,1);  @BND(0,S20S30B23,1);  @BND(0,S20S30B24,1);
@BND(0,S21S28B21,1);  @BND(0,S21S28B22,1);  @BND(0,S21S28B23,1);  @BND(0,S21S28B24,1);
@BND(0,S21S29B21,1);  @BND(0,S21S29B22,1);  @BND(0,S21S29B23,1);  @BND(0,S21S29B24,1);
@BND(0,S21S30B21,1);  @BND(0,S21S30B22,1);  @BND(0,S22S25B21,1);  @BND(0,S22S25B22,1);
@BND(0,S22S26B21,1);  @BND(0,S22S26B22,1);  @BND(0,S22S27B21,1);  @BND(0,S22S27B22,1);
@BND(0,S22S27B23,1);  @BND(0,S22S27B24,1);  @BND(0,S23S25B21,1);  @BND(0,S23S25B22,1);
@BND(0,S23S26B21,1);  @BND(0,S23S26B22,1);  @BND(0,S23S27B21,1);  @BND(0,S23S27B22,1);
@BND(0,S23S27B23,1);  @BND(0,S23S27B24,1);  @BND(0,S24S25B21,1);  @BND(0,S24S25B22,1);
@BND(0,S24S25B23,1);  @BND(0,S24S25B24,1);  @BND(0,S24S26B21,1);  @BND(0,S24S26B22,1);
@BND(0,S24S26B23,1);  @BND(0,S24S26B24,1);  @BND(0,S24S27B21,1);  @BND(0,S24S27B22,1);
@BND(0,S22S28B21,1);  @BND(0,S22S28B22,1);  @BND(0,S22S29B21,1);  @BND(0,S22S29B22,1);
@BND(0,S22S30B21,1);  @BND(0,S22S30B22,1);  @BND(0,S22S30B23,1);  @BND(0,S22S30B24,1);
@BND(0,S23S28B21,1);  @BND(0,S23S28B22,1);  @BND(0,S23S29B21,1);  @BND(0,S23S29B22,1);
@BND(0,S23S30B21,1);  @BND(0,S23S30B22,1);  @BND(0,S23S30B23,1);  @BND(0,S23S30B24,1);
@BND(0,S24S28B21,1);  @BND(0,S24S28B22,1);  @BND(0,S24S28B23,1);  @BND(0,S24S28B24,1);
@BND(0,S24S29B21,1);  @BND(0,S24S29B22,1);  @BND(0,S24S29B23,1);  @BND(0,S24S29B24,1);
@BND(0,S24S30B21,1);  @BND(0,S24S30B22,1);  @BND(0,S37S43B37,1);  @BND(0,S37S43B38,1);
@BND(0,S37S44B37,1);  @BND(0,S37S44B38,1);  @BND(0,S37S45B37,1);  @BND(0,S37S45B38,1);
@BND(0,S37S45B39,1);  @BND(0,S37S45B40,1);  @BND(0,S38S43B37,1);  @BND(0,S38S43B38,1);
@BND(0,S38S44B37,1);  @BND(0,S38S44B38,1);  @BND(0,S38S45B37,1);  @BND(0,S38S45B38,1);
@BND(0,S38S45B39,1);  @BND(0,S38S45B40,1);  @BND(0,S39S43B37,1);  @BND(0,S39S43B38,1);
@BND(0,S39S43B39,1);  @BND(0,S39S43B40,1);  @BND(0,S39S44B37,1);  @BND(0,S39S44B38,1);
@BND(0,S39S44B39,1);  @BND(0,S39S44B40,1);  @BND(0,S39S45B37,1);  @BND(0,S39S45B38,1);
@BND(0,S40S43B37,1);  @BND(0,S40S43B38,1);  @BND(0,S40S44B37,1);  @BND(0,S40S44B38,1);
@BND(0,S40S45B37,1);  @BND(0,S40S45B38,1);  @BND(0,S40S45B39,1);  @BND(0,S40S45B40,1);
@BND(0,S41S43B37,1);  @BND(0,S41S43B38,1);  @BND(0,S41S44B37,1);  @BND(0,S41S44B38,1);
@BND(0,S41S45B37,1);  @BND(0,S41S45B38,1);  @BND(0,S41S45B39,1);  @BND(0,S41S45B40,1);
@BND(0,S42S43B37,1);  @BND(0,S42S43B38,1);  @BND(0,S42S43B39,1);  @BND(0,S42S43B40,1);
@BND(0,S42S44B37,1);  @BND(0,S42S44B38,1);  @BND(0,S42S44B39,1);  @BND(0,S42S44B40,1);
@BND(0,S42S45B37,1);  @BND(0,S42S45B38,1);  @BND(0,S43S49B41,1);  @BND(0,S43S49B42,1);
@BND(0,S43S50B41,1);  @BND(0,S43S50B42,1);  @BND(0,S43S51B41,1);  @BND(0,S43S51B42,1);
@BND(0,S43S51B43,1);  @BND(0,S43S51B44,1);  @BND(0,S44S49B41,1);  @BND(0,S44S49B42,1);
@BND(0,S44S50B41,1);  @BND(0,S44S50B42,1);  @BND(0,S44S51B41,1);  @BND(0,S44S51B42,1);
@BND(0,S44S51B43,1);  @BND(0,S44S51B44,1);  @BND(0,S45S49B41,1);  @BND(0,S45S49B42,1);
@BND(0,S45S49B43,1);  @BND(0,S45S49B44,1);  @BND(0,S45S50B41,1);  @BND(0,S45S50B42,1);
@BND(0,S45S50B43,1);  @BND(0,S45S50B44,1);  @BND(0,S45S51B41,1);  @BND(0,S45S51B42,1);
@BND(0,S43S52B41,1);  @BND(0,S43S52B42,1);  @BND(0,S43S53B41,1);  @BND(0,S43S53B42,1);
@BND(0,S43S54B41,1);  @BND(0,S43S54B42,1);  @BND(0,S43S54B43,1);  @BND(0,S43S54B44,1);
@BND(0,S44S52B41,1);  @BND(0,S44S52B42,1);  @BND(0,S44S53B41,1);  @BND(0,S44S53B42,1);
@BND(0,S44S54B41,1);  @BND(0,S44S54B42,1);  @BND(0,S44S54B43,1);  @BND(0,S44S54B44,1);
@BND(0,S45S52B41,1);  @BND(0,S45S52B42,1);  @BND(0,S45S52B43,1);  @BND(0,S45S52B44,1);
@BND(0,S45S53B41,1);  @BND(0,S45S53B42,1);  @BND(0,S45S53B43,1);  @BND(0,S45S53B44,1);
@BND(0,S45S54B41,1);  @BND(0,S45S54B42,1);  @BND(0,S55S61B53,1);  @BND(0,S55S61B54,1);
@BND(0,S55S62B53,1);  @BND(0,S55S62B54,1);  @BND(0,S55S63B53,1);  @BND(0,S55S63B54,1);
@BND(0,S55S63B55,1);  @BND(0,S55S63B56,1);  @BND(0,S56S61B53,1);  @BND(0,S56S61B54,1);
@BND(0,S56S62B53,1);  @BND(0,S56S62B54,1);  @BND(0,S56S63B53,1);  @BND(0,S56S63B54,1);
@BND(0,S56S63B55,1);  @BND(0,S56S63B56,1);  @BND(0,S57S61B53,1);  @BND(0,S57S61B54,1);
@BND(0,S57S61B55,1);  @BND(0,S57S61B56,1);  @BND(0,S57S62B53,1);  @BND(0,S57S62B54,1);
@BND(0,S57S62B55,1);  @BND(0,S57S62B56,1);  @BND(0,S57S63B53,1);  @BND(0,S57S63B54,1);
@BND(0,S58S61B53,1);  @BND(0,S58S61B54,1);  @BND(0,S58S62B53,1);  @BND(0,S58S62B54,1);
@BND(0,S58S63B53,1);  @BND(0,S58S63B54,1);  @BND(0,S58S63B55,1);  @BND(0,S58S63B56,1);
@BND(0,S59S61B53,1);  @BND(0,S59S61B54,1);  @BND(0,S59S62B53,1);  @BND(0,S59S62B54,1);
@BND(0,S59S63B53,1);  @BND(0,S59S63B54,1);  @BND(0,S59S63B55,1);  @BND(0,S59S63B56,1);
@BND(0,S60S61B53,1);  @BND(0,S60S61B54,1);  @BND(0,S60S61B55,1);  @BND(0,S60S61B56,1);
@BND(0,S60S62B53,1);  @BND(0,S60S62B54,1);  @BND(0,S60S62B55,1);  @BND(0,S60S62B56,1);
@BND(0,S60S63B53,1);  @BND(0,S60S63B54,1);
```

```
!Declare the S Variable as Binary Integer Variables;
@BIN(S1111);  @BIN(S1112);  @BIN(S1113);  @BIN(S1121);  @BIN(S1122);  @BIN(S1123);
@BIN(S1211);  @BIN(S1212);  @BIN(S1213);  @BIN(S1311);  @BIN(S1312);  @BIN(S1313);
@BIN(S1321);  @BIN(S1322);  @BIN(S1323);  @BIN(S2111);  @BIN(S2112);  @BIN(S2113);
@BIN(S2121);  @BIN(S2122);  @BIN(S2123);  @BIN(S2211);  @BIN(S2212);  @BIN(S2213);
@BIN(S2221);  @BIN(S2222);  @BIN(S2223);  @BIN(S3111);  @BIN(S3112);  @BIN(S3113);
@BIN(S3121);  @BIN(S3122);  @BIN(S3123);  @BIN(S3211);  @BIN(S3212);  @BIN(S3213);
@BIN(S3311);  @BIN(S3312);  @BIN(S3313);  @BIN(S3321);  @BIN(S3322);  @BIN(S3323);
@BIN(S4111);  @BIN(S4112);  @BIN(S4113);  @BIN(S4121);  @BIN(S4122);  @BIN(S4123);
@BIN(S4211);  @BIN(S4212);  @BIN(S4213);

!Declare the B Variable as Binary Integer Variables;
@BIN(B1111);  @BIN(B1112);  @BIN(B1121);  @BIN(B1122);  @BIN(B1211);  @BIN(B1212);
@BIN(B1221);  @BIN(B1222);  @BIN(B1311);  @BIN(B1312);  @BIN(B1321);  @BIN(B1322);
@BIN(B1411);  @BIN(B1412);  @BIN(B1421);  @BIN(B1422);  @BIN(B2111);  @BIN(B2112);
@BIN(B2121);  @BIN(B2122);  @BIN(B2211);  @BIN(B2212);  @BIN(B2221);  @BIN(B2222);
@BIN(B2311);  @BIN(B2312);  @BIN(B2321);  @BIN(B2322);  @BIN(B3111);  @BIN(B3112);
@BIN(B3121);  @BIN(B3122);  @BIN(B3211);  @BIN(B3212);  @BIN(B3221);  @BIN(B3222);
@BIN(B3311);  @BIN(B3312);  @BIN(B3321);  @BIN(B3322);  @BIN(B3411);  @BIN(B3412);
@BIN(B3421);  @BIN(B3422);  @BIN(B4111);  @BIN(B4112);  @BIN(B4121);  @BIN(B4122);
@BIN(B4211);  @BIN(B4212);  @BIN(B4221);  @BIN(B4222);  @BIN(B4311);  @BIN(B4312);
@BIN(B4321);  @BIN(B4322);

!Declare the X Variable as Binary Integer Variables;
@BIN(X111);  @BIN(X112);  @BIN(X113);  @BIN(X121);  @BIN(X122);  @BIN(X123);
@BIN(X211);  @BIN(X212);  @BIN(X213);  @BIN(X221);  @BIN(X222);  @BIN(X223);
@BIN(X311);  @BIN(X312);  @BIN(X313);
@BIN(X411);  @BIN(X412);  @BIN(X413);  @BIN(X421);  @BIN(X422);  @BIN(X423);
@BIN(X511);  @BIN(X512);  @BIN(X513);

END
!End of MODEL;
```

APPENDIX B

EXPERIMENTAL DATA

APPENDIX B.1

Experimental Data for Small Problem

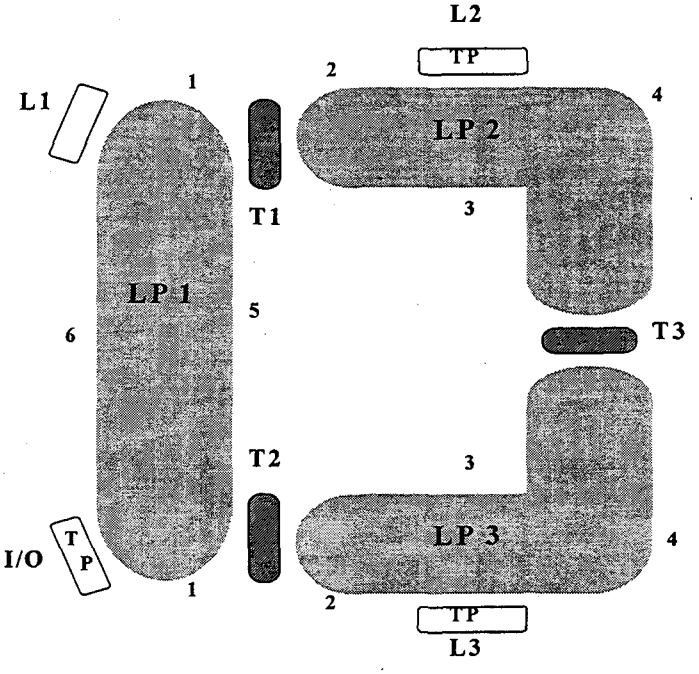

Figure B.1 AGV layout for small problem

Table B.1 AGV capacity for each small problem instance

Problem Instance	Unlimited AGV Capacity AGV 1, 2, and 3	Limited AGV Capacity AGV 1, 2, and 3
1	10000	578
2	10000	482
3	10000	500,180,500
4	10000	584
5	10000	700
6	10000	392
7	10000	480
8	10000	479
9	10000	427
10	10000	510

Table B.2 Part production plan for each small problem instance

Problem Instance	Part No.	Machine Type Assignment			Processing Time			Batch
		O1	O2	O3	O1	O2	O3	
1	P1	1	3	4	4.0	5.0	6.0	3
	P2	2	4	-	3.5	4.5	-	2
	P3	2	5	2	4.0	5.0	6.0	3
	P4	1	3	-	7.0	3.0	-	2
2	P1	4	2	-	5.9	6.2	-	2
	P2	5	4	3	6.5	4.9	1.0	3
	P3	4	1	3	0.9	4.4	7.0	2
	P4	1	2	5	5.8	2.7	1.9	3
3	P1	1	3	2	5.0	6.2	2.9	3
	P2	4	3	-	2.7	5.5	-	3
	P3	4	2	4	1.1	6.1	7.7	3
	P4	4	5	-	1.6	3.4	-	2
4	P1	4	2	-	1.2	7.3	-	3
	P2	5	3	-	4.8	0.5	-	3
	P3	1	5	3	3.0	6.0	6.6	2
	P4	3	4	3	7.2	4.8	7.3	3
5	P1	5	4	-	4.4	0.5	-	2
	P2	3	2	4	6.2	7.2	7.8	3
	P3	3	4	2	7.4	0.9	1.9	2
	P4	1	4	3	1.4	5.2	4.7	3
6	P1	3	1	-	2.7	2.0	-	2
	P2	3	4	-	0.9	3.6	-	3
	P3	5	1	-	7.8	4.0	-	3
	P4	1	2	1	7.1	1.6	6.9	2
7	P1	1	4	5	0.9	5.8	5.3	2
	P2	5	1	3	2.4	6.8	3.1	2
	P3	3	4	-	4.6	2.3	-	2
	P4	2	1	2	5.6	3.6	5.3	3
8	P1	5	3	1	6.3	2.7	4.3	2
	P2	1	2	5	7.6	3.3	7.2	3
	P3	1	4	2	5.7	3.2	1.6	3
	P4	2	1	-	0.5	1.6	-	2
9	P1	5	3	-	2.0	3.7	-	2
	P2	4	3	-	5.2	4.0	-	3
	P3	2	1	-	1.7	3.6	-	2
	P4	1	2	3	7.9	5.4	3.6	3
10	P1	4	5	2	0.8	4.5	1.9	2
	P2	2	1	-	2.8	2.6	-	3
	P3	4	1	-	5.3	6.7	-	3
	P4	3	2	3	7.5	5.0	5.4	2

APPENDIX B.2

Experimental Data for Medium Problem

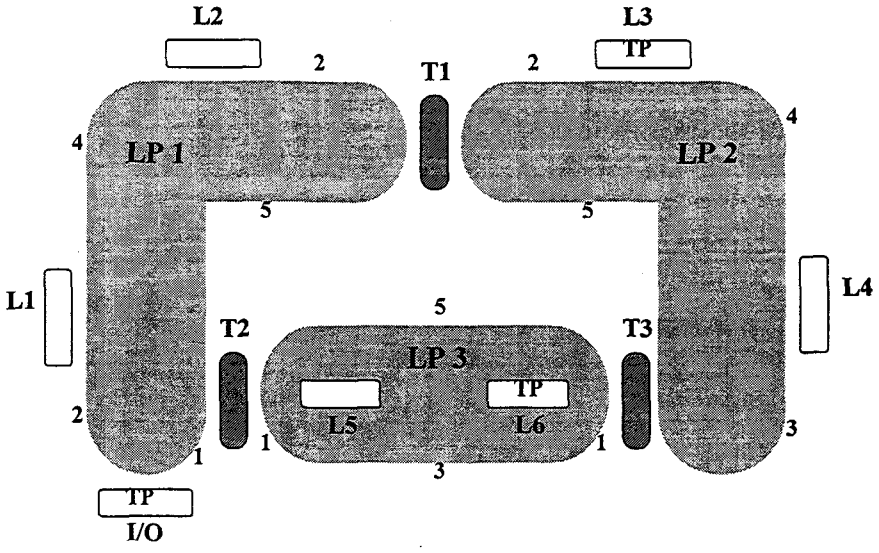

Figure B2. AGV layout for medium problem

Table B.3 AGV capacity for each medium problem instance

Problem Instance	Unlimited AGV Capacity	Limited AGV Capacity	
	AGV 1, 2, and 3	AGV 1	AGV 2 and 3
1	15000	2400	15000
2	15000	2900	15000
3	15000	3433	15000
4	15000	2250	15000
5	15000	2595	15000

Table B.4 Part production plan for each medium problem instance

Problem Instance 1

Part No.	Machine Type Assignment				Processing Time				Batch
	O1	O2	O3	O4	O1	O2	O3	O4	
P1	2	9	2	-	2.5	2.4	3.5	-	4
P2	1	3	4	-	2.8	1.8	2.9	-	5
P3	6	2	-	-	1.7	0.8	-	-	1
P4	8	7	10	5	1.3	3.7	3.7	3.2	3
P5	7	8	3	9	3.6	0.5	3.5	4.0	1
P6	7	5	3	-	3.1	3.1	3.3	-	5
P7	9	2	4	7	3.3	2.9	1.1	3.5	1
P8	8	10	-	-	2.3	1.7	-	-	2
P9	9	6	-	-	2.5	3.0	-	-	2
P10	8	4	-	-	2.5	2.7	-	-	3
P11	10	1	-	-	3.8	2.3	-	-	2
P12	10	4	7	-	3.4	2.2	3.0	-	5
P13	5	10	-	-	1.8	4.0	-	-	3
P14	8	5	3	-	3.4	3.2	1.2	-	2
P15	9	5	4	7	1.4	1.7	3.8	2.1	4
P16	9	8	-	-	2.1	1.6	-	-	4
P17	6	8	-	-	2.5	2.4	-	-	1
P18	8	3	-	-	3.0	2.0	-	-	5
P19	6	1	-	-	3.4	3.2	-	-	2
P20	2	9	-	-	1.4	3.3	-	-	2

Problem Instance 2

Part No.	Machine Type Assignment				Processing Time				Batch
	O1	O2	O3	O4	O1	O2	O3	O4	
P1	5	9	-	-	3.2	2.9	-	-	5
P2	6	5	8	-	2.5	2.1	2.1	-	4
P3	5	10	3	9	1.3	0.9	0.8	3.7	2
P4	3	6	-	-	2.4	3.6	-	-	5
P5	8	6	7	-	3.8	3.7	3.7	-	3
P6	3	8	-	-	2.8	1.9	-	-	3
P7	10	8	7	-	3.0	3.1	2.6	-	5
P8	7	6	-	-	1.7	3.6	-	-	3
P9	7	6	2	-	2.1	3.6	3.6	-	4
P10	9	1	-	-	2.0	2.0	-	-	3
P11	8	6	3	-	1.3	2.1	3.9	-	2
P12	10	4	-	-	1.8	2.0	-	-	3
P13	1	3	-	-	3.0	3.5	-	-	1
P14	6	2	6	7	1.9	3.3	2.7	3.5	4
P15	5	8	5	-	1.8	3.3	2.5	-	2
P16	1	5	3	-	3.2	1.9	4.0	-	4
P17	2	5	3	9	3.9	1.3	2.5	3.9	3
P18	10	9	-	-	3.1	1.7	-	-	5
P19	7	6	2	1	2.8	1.8	3.1	1.8	4
P20	7	1	-	-	3.4	2.7	-	-	3

Problem Instance 3

Part No.	Machine Type Assignment				Processing Time				Batch
	O1	O2	O3	O4	O1	O2	O3	O4	
P1	2	9	-	-	3.7	3.3	-	-	1
P2	5	9	-	-	2.3	1.3	-	-	2
P3	2	5	7	2	3.1	1.9	3.8	1.7	4
P4	1	8	1	6	0.8	1.3	0.8	1.4	2
P5	6	4	-	-	1.4	3.9	-	-	1
P6	10	8	-	-	3.0	2.2	-	-	4
P7	9	10	1	8	3.1	2.8	1.0	2.0	2
P8	9	6	7	-	1.9	1.4	3.5	-	3
P9	3	5	-	-	2.1	3.8	-	-	5
P10	2	5	-	-	3.0	3.1	-	-	3
P11	5	4	3	4	1.9	2.2	2.4	2.6	5
P12	2	7	4	8	1.7	1.2	3.5	3.2	2
P13	4	1	-	-	2.4	2.9	-	-	5
P14	7	4	8	4	3.7	1.8	4.0	3.7	5
P15	7	5	-	-	4.0	3.5	-	-	5
P16	5	2	-	-	2.5	2.7	-	-	5
P17	2	1	8	3	1.7	3.4	2.2	1.8	5
P18	10	7	4	9	1.9	1.4	2.6	1.3	3
P19	7	4	-	-	3.8	1.8	-	-	4
P20	4	1	-	-	2.9	3.6	-	-	5

Problem Instance 4

Part No.	Machine Type Assignment				Processing Time				Batch
	O1	O2	O3	O4	O1	O2	O3	O4	
P1	1	2	-	-	3.1	2.1	-	-	2
P2	1	7	-	-	1.6	3.3	-	-	4
P3	8	1	9	-	2.9	0.5	0.8	-	1
P4	4	1	8	-	2.6	1.9	1.8	-	4
P5	4	7	-	-	1.5	1.4	-	-	1
P6	6	5	-	-	3.3	3.1	-	-	4
P7	5	3	-	-	2.4	0.9	-	-	1
P8	3	1	8	-	2.6	3.0	3.5	-	5
P9	5	10	3	5	3.1	2.4	3.8	3.4	2
P10	8	1	10	-	2.2	3.7	2.2	-	3
P11	8	5	7	-	1.1	3.8	3.8	-	3
P12	7	5	6	-	2.3	3.3	2.4	-	5
P13	4	5	-	-	1.8	1.7	-	-	4
P14	1	8	7	-	1.8	3.1	2.2	-	5
P15	7	4	10	-	1.1	3.0	1.8	-	1
P16	6	10	-	-	1.5	1.4	-	-	3
P17	9	10	-	-	3.1	3.8	-	-	2
P18	8	9	-	-	2.8	2.9	-	-	4
P19	4	1	-	-	3.5	2.8	-	-	5
P20	9	3	6	7	0.7	2.8	3.0	1.0	2

Problem Instance 5

Part No.	Machine Type Assignment				Processing Time				Batch
	O1	O2	O3	O4	O1	O2	O3	O4	
P1	4	9	-	-	3.2	3.4	-	-	4
P2	10	7	-	-	1.9	3.6	-	-	4
P3	3	4	7	-	3.1	1.3	4.0	-	1
P4	10	4	7	-	1.5	1.0	2.2	-	1
P5	2	7	4	-	3.5	1.7	3.6	-	5
P6	6	9	-	-	3.3	2.6	-	-	3
P7	10	8	-	-	3.0	3.9	-	-	5
P8	8	4	2	10	2.2	2.4	2.5	3.2	1
P9	1	2	-	-	1.5	3.8	-	-	3
P10	1	5	8	1	1.6	2.3	2.0	3.1	1
P11	5	10	3	-	1.7	0.9	0.9	-	2
P12	4	1	5	7	1.7	2.6	3.8	2.5	5
P13	10	8	10	-	0.8	2.2	1.8	-	1
P14	2	6	-	-	2.9	1.7	-	-	3
P15	7	4	10	-	1.9	1.7	3.3	-	5
P16	1	4	-	-	3.5	2.2	-	-	4
P17	6	2	10	-	2.9	2.6	2.2	-	4
P18	6	7	-	-	2.6	1.9	-	-	5
P19	8	10	5	7	1.4	2.8	3.3	1.5	3
P20	8	5	-	-	1.0	1.9	-	-	3

APPENDIX B.3

Experimental Data for Large Problem

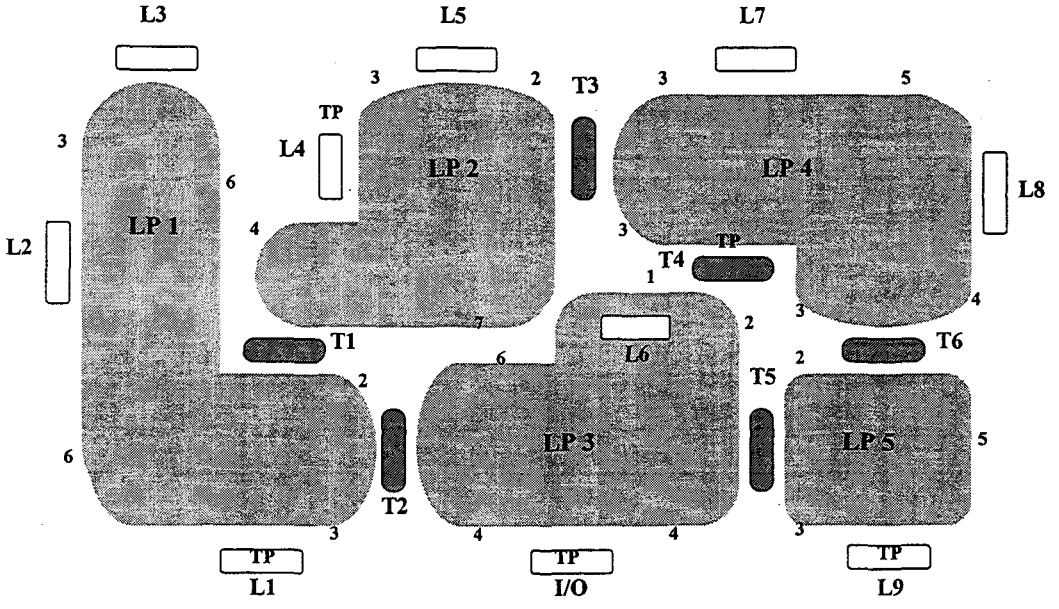

Figure B3. AGV layout for large problem
Note: All of the AGV capacity for the large problem instances is set to 20000.

Table B.5 Part production plan for each large problem instance

Problem Instance 1

Part No.	Machine Type Assignment					Processing Time					Batch
	O1	O2	O3	O4	O5	O1	O2	O3	O4	O5	
P1	13	9	10	2	-	1.7	1.2	1.5	1.1	-	2
P2	3	12	-	-	-	3.6	0.9	-	-	-	2
P3	1	6	3	-	-	4	1.1	1.4	-	-	2
P4	11	12	-	-	-	3.8	1.2	-	-	-	2
P5	13	3	4	9	6	3.4	1.8	1.2	3.6	1.8	3
P6	8	2	-	-	-	2.1	2.1	-	-	-	5
P7	15	4	-	-	-	1	3	-	-	-	3
P8	13	9	12	6	-	4	3.5	2.2	3.8	-	2
P9	1	6	-	-	-	3.2	2.6	-	-	-	6
P10	15	14	13	11	-	1.9	1.7	2.9	3.9	-	5
P11	11	7	-	-	-	2.5	0.5	-	-	-	1
P12	2	8	-	-	-	3.3	2	-	-	-	6
P13	7	13	-	-	-	3.4	0.5	-	-	-	1
P14	9	2	3	4	-	2	3.5	1.9	1.5	-	3
P15	8	5	6	-	-	3	3.1	1.8	-	-	5
P16	10	7	5	10	-	3.6	2.8	2.6	1.9	-	5
P17	12	9	-	-	-	4	0.7	-	-	-	1
P18	4	9	15	10	-	2.8	3.6	2.7	1.1	-	3
P19	11	2	8	-	-	3.8	2.7	3.2	-	-	5
P20	11	9	5	-	-	3.1	2.5	3.2	-	-	5
P21	12	7	11	-	-	2.8	3.3	2.8	-	-	6
P22	9	5	-	-	-	2.3	3.1	-	-	-	1
P23	5	4	-	-	-	1.8	3	-	-	-	2
P24	3	13	-	-	-	1.9	3	-	-	-	5
P25	4	14	13	-	-	1.9	2	2.5	-	-	2
P26	14	10	-	-	-	2.3	2.2	-	-	-	1
P27	11	3	4	-	-	1.6	1.5	2.7	-	-	2
P28	7	11	-	-	-	2.1	1.7	-	-	-	5
P29	10	6	2	5	12	1.4	1	3.2	0.7	3.2	2
P30	2	4	11	-	-	2.8	1.9	2.1	-	-	3

Problem Instance 2

Part No.	Machine Type Assignment					Processing Time					Batch
	O1	O2	O3	O4	O5	O1	O2	O3	O4	O5	
P1	12	8	3	-	-	2.9	4	3.8	-	-	1
P2	14	12	-	-	-	3	2.8	-	-	-	3
P3	7	14	13	-	-	1.1	1.8	1.9	-	-	3
P4	11	8	7	11	-	2.3	0.8	2.1	2.3	-	1
P5	5	13	-	-	-	3.2	3.1	-	-	-	5
P6	2	11	1	-	-	2	2.1	2	-	-	5
P7	2	7	-	-	-	3.6	3.4	-	-	-	1
P8	9	6	8	6	-	3.2	3.5	2.1	2.2	-	5
P9	11	3	10	4	15	1	3.3	2.8	3	1.2	3
P10	8	3	-	-	-	1.4	1.2	-	-	-	2
P11	2	6	11	3	-	3.8	3.9	3.3	2.9	-	2
P12	4	10	-	-	-	3.3	0.5	-	-	-	1
P13	10	3	-	-	-	1.1	2.3	-	-	-	2
P14	6	1	3	1	-	3.4	2.8	2.3	3.4	-	5
P15	10	15	-	-	-	2.3	3.6	-	-	-	2
P16	15	1	4	-	-	3.3	1.3	1.9	-	-	2
P17	8	14	3	-	-	2.4	1.9	2.6	-	-	5
P18	14	10	-	-	-	3.8	3.2	-	-	-	1
P19	11	10	1	-	-	2.9	2	1.5	-	-	3
P20	4	2	-	-	-	2.9	3	-	-	-	5
P21	10	14	8	-	-	0.7	2.1	2.8	-	-	2
P22	8	13	-	-	-	3	1.5	-	-	-	2
P23	12	9	-	-	-	2	3.5	-	-	-	6
P24	4	8	2	10	-	2.1	1.8	2.9	3	-	4
P25	14	12	-	-	-	2.2	4	-	-	-	1
P26	12	13	8	1	10	0.7	0.7	2.1	2.2	1.9	2
P27	10	15	7	-	-	2	2.6	2	-	-	5
P28	12	11	2	6	-	2.8	1.9	3	0.7	-	2
P29	9	4	-	-	-	1.1	2.1	-	-	-	3
P30	2	13	-	-	-	1.8	2.5	-	-	-	2

Problem Instance 3

Part No.	Machine Type Assignment					Processing Time					Batch
	O1	O2	O3	O4	O5	O1	O2	O3	O4	O5	
P1	8	6	-	-	-	2.8	2.4	-	-	-	5
P2	14	12	13	-	-	3.5	2.8	3.4	-	-	3
P3	9	2	-	-	-	3.5	2.1	-	-	-	4
P4	2	1	7	8	-	2.9	3.9	1.8	3.5	-	5
P5	10	15	-	-	-	1.7	1.5	-	-	-	4
P6	2	14	7	-	-	3	1.4	1.9	-	-	4
P7	7	6	8	10	-	3.3	1.8	1.3	1.2	-	3
P8	1	12	-	-	-	2.9	3.1	-	-	-	5
P9	6	3	-	-	-	1.2	2.9	-	-	-	3
P10	7	2	-	-	-	1.9	1.8	-	-	-	1
P11	5	2	5	8	6	0.9	3.1	0.6	1.5	1	1
P12	4	12	-	-	-	2.8	1.3	-	-	-	1
P13	1	11	3	14	-	2	3.8	2.1	2.6	-	4
P14	15	6	3	6	-	1.8	1.7	4	3.6	-	4
P15	11	9	15	9	10	2.1	3.2	2.4	2.1	2.4	6
P16	13	3	11	-	-	3.2	2.2	2	-	-	6
P17	14	8	-	-	-	1.4	2.1	-	-	-	4
P18	3	11	8	3	-	2.7	3.7	2.7	3.9	-	6
P19	14	13	-	-	-	2.2	4	-	-	-	6
P20	12	3	15	-	-	0.5	3.3	3.1	-	-	1
P21	6	12	-	-	-	3.2	2.1	-	-	-	6
P22	6	8	-	-	-	2	2.4	-	-	-	3
P23	5	13	8	-	-	1.1	1.4	3.3	-	-	3
P24	12	2	10	6	15	2.5	3	2.3	1.7	2.5	5
P25	11	13	9	-	-	2.8	3.5	2.5	-	-	5
P26	4	2	-	-	-	3.9	3.9	-	-	-	4
P27	9	8	5	-	-	2.1	2.2	2	-	-	4
P28	1	2	14	-	-	2.2	3.1	2.8	-	-	6
P29	4	1	10	5	-	3.1	2.8	2.8	2	-	6
P30	6	4	-	-	-	1.1	3.2	-	-	-	1

Problem Instance 4

Part No.	Machine Type Assignment					Processing Time					Batch
	O1	O2	O3	O4	O5	O1	O2	O3	O4	O5	
P1	12	1	15	1	-	3.4	2.3	3.7	2.5	-	6
P2	9	12	15	1	-	2.9	2.9	3	2.3	-	5
P3	15	14	-	-	-	2.4	3.9	-	-	-	5
P4	11	9	4	8	-	2.6	2.3	2.2	1.1	-	3
P5	4	10	1	4	-	3.7	1	1.5	1.1	-	3
P6	12	13	-	-	-	4	1.5	-	-	-	3
P7	10	9	6	-	-	1	1.1	3	-	-	3
P8	5	4	-	-	-	3.4	3.8	-	-	-	3
P9	13	9	-	-	-	2.4	1.7	-	-	-	4
P10	11	10	-	-	-	3.8	4	-	-	-	2
P11	5	1	-	-	-	1.1	1.4	-	-	-	3
P12	6	7	-	-	-	4	3.2	-	-	-	3
P13	15	12	7	-	-	2.8	2.1	3.3	-	-	4
P14	5	6	4	-	-	3.7	2	3.2	-	-	1
P15	8	12	11	-	-	3.6	2.5	1.5	-	-	2
P16	9	10	14	-	-	0.8	1.6	0.8	-	-	1
P17	15	5	-	-	-	2.2	1.9	-	-	-	3
P18	12	2	5	15	-	1.2	2.1	4	2.8	-	2
P19	11	6	-	-	-	1.6	2.9	-	-	-	1
P20	11	6	5	11	-	1.8	1.4	2.2	2	-	2
P21	6	5	4	3	6	2	3.2	2.6	3.7	2.6	4
P22	10	1	-	-	-	3.2	2.2	-	-	-	4
P23	9	11	-	-	-	1.6	3.8	-	-	-	1
P24	5	11	7	-	-	2	2.7	1.7	-	-	5
P25	9	5	2	15	-	2.2	2.3	2.3	2.6	-	6
P26	8	6	-	-	-	3.9	3.2	-	-	-	2
P27	13	10	-	-	-	2.6	2.6	-	-	-	6
P28	5	11	7	-	-	1.8	2.9	1.9	-	-	5
P29	11	13	1	-	-	1	3	2.8	-	-	2
P30	14	2	8	-	-	2.7	3.3	3.7	-	-	5

Problem Instance 5

Part No.	Machine Type Assignment					Processing Time					Batch
	O1	O2	O3	O4	O5	O1	O2	O3	O4	O5	
P1	1	8	14	-	-	1.8	2.5	2.9	-	-	4
P2	12	5	-	-	-	1.6	3.6	-	-	-	1
P3	15	5	15	-	-	3.9	3	2	-	-	6
P4	4	3	-	-	-	1.8	2.2	-	-	-	3
P5	1	10	8	-	-	3.7	3.8	1.6	-	-	4
P6	11	10	-	-	-	1	2.8	-	-	-	2
P7	1	15	-	-	-	3.4	3.9	-	-	-	3
P8	9	13	-	-	-	2	2.7	-	-	-	5
P9	6	7	-	-	-	1.8	2.7	-	-	-	2
P10	5	12	2	15	14	1.7	2.3	3.9	1	2.8	2
P11	6	3	-	-	-	3.8	3.1	-	-	-	4
P12	8	4	14	5	-	4	3.8	3.6	3.2	-	4
P13	1	13	-	-	-	2	2.4	-	-	-	5
P14	11	2	3	9	-	1.9	1	3	2.7	-	1
P15	9	13	14	-	-	4	3.7	2	-	-	6
P16	1	15	3	-	-	2.7	3	1.7	-	-	5
P17	13	6	15	-	-	1.1	1.4	2.3	-	-	1
P18	10	6	7	-	-	3.2	2.8	2.6	-	-	2
P19	12	7	-	-	-	1.4	1.8	-	-	-	4
P20	9	1	6	12	2	3.3	2.9	1.8	2.9	3.1	4
P21	7	11	-	-	-	3.1	1.3	-	-	-	1
P22	5	4	1	-	-	3.2	0.5	2.4	-	-	1
P23	15	8	15	1	-	3.4	1.4	2.7	3.2	-	4
P24	3	15	9	-	-	2.3	1	1.8	-	-	3
P25	14	1	-	-	-	1.6	1.8	-	-	-	2
P26	15	14	8	4	13	2.7	1.2	1.9	1.3	0.9	2
P27	3	1	12	15	11	2.4	0.9	3.5	0.8	3	2
P28	3	2	15	1	-	2.4	3.9	4	1.2	-	1
P29	10	9	12	13	1	3.8	2.7	3.2	1.8	2.6	5
P30	2	8	9	11	6	2.9	3.2	1.1	0.8	0.6	1

APPENDIX B.4

Tabu Search Parameters

Table B.6 Parameters used in tabu search-based heuristics for each problem structure

Parameter		5P*3O*5MT 3L*3A*2R		20P*4O*10MT* 6L*3A*2R		30P*5O*15MT* 9L*5A*3R	
		Inside Search	Outside Search	Inside Search	Outside Search	Inside Search	Outside Search
Number of Neighborhood Search		18	27	144	240	252	576
Tabu List Size	Fixed	1	2	4	7	5	11
	Variable						
	- Initial	1	2	4	7	5	11
	- Decrease	1	1	3	5	4	8
	- Increase	2	3	5	9	7	14
Number of Iterations w/o Improvement	Fixed	2	3	4	6	5	7
	Variable	1	2	2	3	3	4
Number of Entries into Index List		3	6	8	19	10	38
Number of Machines Fix to Locations		-	1	-	3	-	4
Number of Long Term Restarts		-	1	-	2	-	2

APPENDIX C

NORMAL PROBABILITY PLOTS

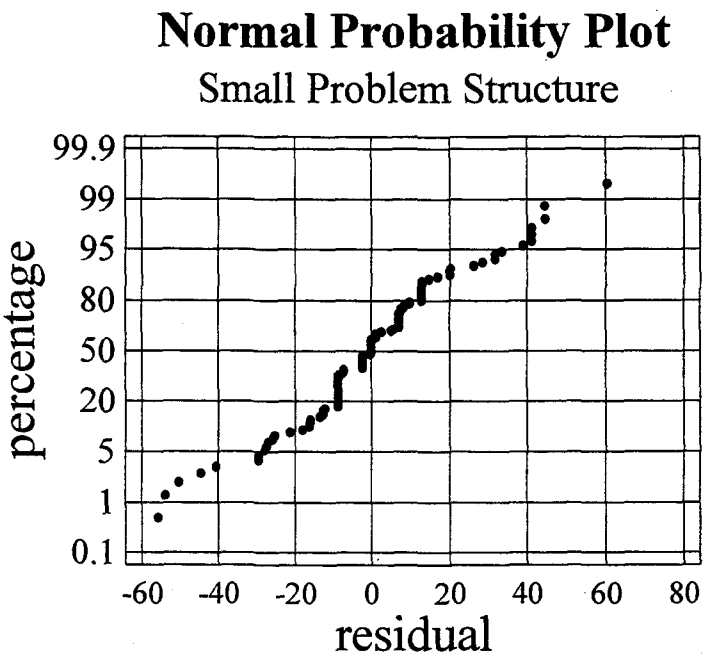

Figure C.1 Normal probability plot for small problem

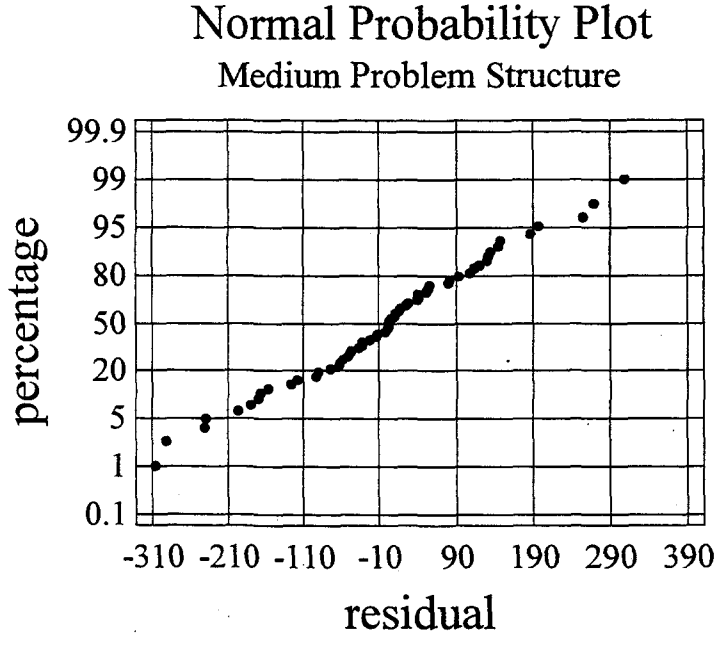

Figure C.2 Normal probability plot for medium problem

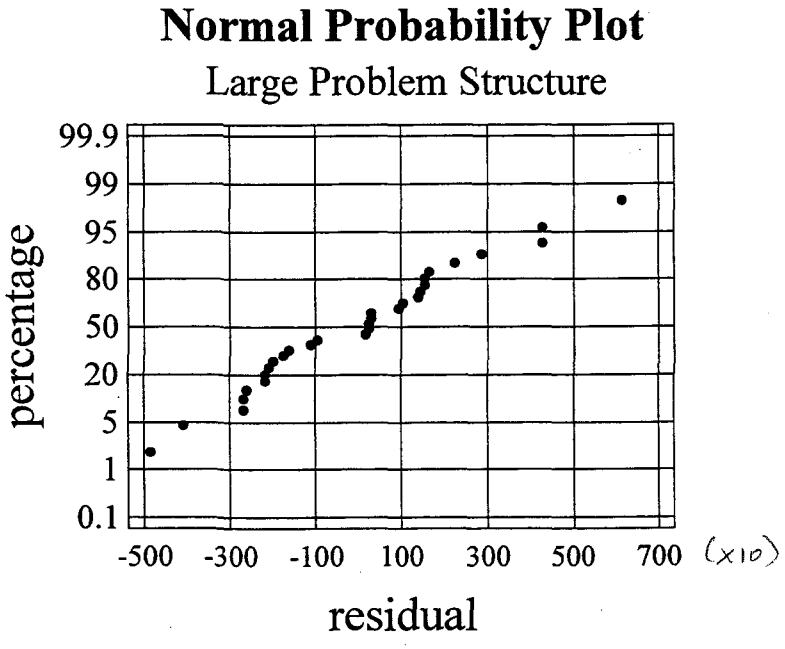

Figure C.3 Normal probability plot for large problem

APPENDIX D

EXPERIMENTAL RESULTS

Table D.1 Results obtained for small problem based on unlimited AGV capacity

Problem Instance	TS1		TS2		TS3		TS4		TS5		TS6	
	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time
1	861	12	861	21	831	21	861	9	861	21	831	20
2	785	12	785	22	785	19	800	16	800	29	800	29
3	879	6	879	10	834	8	813	14	813	26	813	26
4	889	5	889	9	889	12	889	12	889	23	889	32
5	951	15	951	27	951	24	951	13	951	25	951	27
6	699	10	699	16	699	13	699	10	699	17	699	17
7	723	7	723	11	723	12	723	10	723	18	723	19
8	952	3	952	6	952	7	845	17	845	27	845	27
9	763	6	763	12	677	19	763	16	763	29	677	29
10	777	20	777	26	777	34	839	12	839	21	777	24

Table D.2 Results obtained for small problem based on limited AGV capacity

Problem Instance	TS1		TS2		TS3		TS4		TS5		TS6	
	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time
1	885	13	885	20	855	21	885	15	885	24	855	25
2	860	22	860	30	809	35	860	22	860	39	809	41
3	815	17	815	32	815	22	858	13	815	26	815	26
4	924	7	924	14	924	13	924	14	924	26	924	26
5	993	19	993	36	993	25	993	13	993	24	993	23
6	723	12	723	17	723	16	723	13	723	20	723	19
7	747	12	747	21	747	20	747	16	747	25	747	30
8	877	8	877	14	877	15	877	17	877	32	877	33
9	863	8	863	15	788	21	788	20	788	36	788	30
10	809	9	809	16	809	14	809	14	809	25	809	24

Table D.3 Results obtained for medium problem based on unlimited AGV capacity

Problem Instance	TS1		TS2		TS3		TS4		TS5		TS6	
	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time
1	4108	6264	4108	15740	4108	15390	4364	4490	4256	11482	4219	10703
2	5529	3768	5387	11622	5300	13957	5545	2237	5545	7246	5419	8251
3	5395	3438	5395	12034	5395	8314	5707	2226	5707	5864	5312	7091
4	3992	3829	3992	5510	3992	8436	4031	1954	3984	7304	4031	6098
5	3991	5900	3991	13947	3991	15769	4411	1788	4411	5167	4411	6648

Table D.4 Results obtained for medium problem based on limited AGV capacity

Problem Instance	TS1		TS2		TS3		TS4		TS5		TS6	
	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time
1	4157	7076	4157	13522	4157	19410	4239	4963	4239	19259	4239	9874
2	5513	5290	5513	17290	5208	38398	5672	7883	5672	19144	5672	37195
3	5396	9818	5396	41801	5396	18300	5836	18557	5836	38035	5836	28973
4	4556	1634	4556	4421	4556	10060	4556	3890	4556	10711	4556	9243
5	4908	1439	4908	4132	4908	13311	4475	4047	4475	10111	4475	11336

Table D.5 Results obtained for large problem based on unlimited AGV capacity

Problem Instance	TS1		TS2		TS3		TS4		TS5		TS6	
	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time
1	16416	9877	15219	77423	16416	62800	15619	27931	15619	103332	15310	75645
2	13237	22384	13054	69149	13237	69260	13395	25445	13395	76503	13395	68946
3	19817	36294	19817	161358	19817	156519	19329	29852	19329	96108	19329	103789
4	15938	10556	14833	69392	15938	113787	15809	25823	15809	70893	15809	87172
5	14587	16075	14118	93049	14587	106189	14825	42795	14564	151831	14825	78816

APPENDIX E

ANALYSIS OF VARIANCE FOR RANDOMIZED BLOCK DESIGN EXPERIMENT

Table E.1 Results obtained from the analysis of variance for small problem

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-Ratio	P-value
Main Effect: Heuristic	7433.07	5	1486.61	3.12	0.0118
Blocking: Test Problem	734872.00	19	38677.50	81.25	0.0000
Residual	45224.90	95	476.052		
Total (Corrected)	787530.00	119			

All F-ratios are based on the residual mean square error.

Table E.2 Results obtained from the analysis of variance for medium problem

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-Ratio	P-value
Main Effect: Heuristic	271223	5	54244.5	2.44	0.0488
Blocking: Test Problem	23442400	9	2604710	117.06	0.0000
Residual	1001290	45	22251		
Total (Corrected)	24714900	59			

All F-ratios are based on the residual mean square error.

Table E.3 Results obtained from the analysis of variance for large problem

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-Ratio	P-value
Main Effect: Heuristic	1189510	5	237901	2.52	0.0636
Blocking: Test Problem	132284000	4	33071100	349.77	0.0000
Residual	1891010	20	94550.6		
Total (Corrected)	135365000	29			

All F-ratios are based on the residual mean square error.

APPENDIX F

PSEUDO CODE FOR
TABU SEARCH-BASED HEURISTIC ALGORITHM

MAIN PROGRAM-OUTSIDE SEARCH

```
Generate the initial Part-Machine Assignment
Generate the initial Machine-Location Identification
Determine the travel distance and the AGV routes between any two locations.
Determine the tabu search parameters used for the outside and inside search
Do
{
    Initialize the Outside Candidate List (OCL) and the Outside Index List (OIL)
    Initialize the frequency matrix used for Outside Long Term Memory (OLTM)
    Initialize the Outside Tabu List (OTL)
    Evaluate the total service time of the initial solution.
    Call subroutine (INSIDE SEARCH) with the initial outside configuration
    Insert the initial machine-location configuration into OCL and OIL.
    Set the Aspiration Value (out_AL) to the value returned by the inside search.
    Update the OLTM matrix
    Set the initial machine location (outside) configuration with its best part-machine configuration
    (inside) as the current parent node.
    Do
    {
        For each seed generated from the current parent node
        {
            The best outside solution ← large number
            If (seed ∈ OCL), Skip it.
            Evaluate the seed
            If (outside move ≠ tabu) or (outside move = tabu but out_AL is satisfied)
                If (seed < the best outside solution)
                {
                    OCL ← current seed
                    OTL ← current move
                    The best outside solution ← current seed
                }
        }
        Call subroutine (INSIDE SEARCH) with the current best outside solution
        Next parent node ← current best outside + current best inside solution
        If (next parent node < out_AL), Update out_AL
        If (next parent node = local optima)
        {
            OIL ← current best outside configuration
            Entries into Outside Index List (EOIL) is increased by 1
        }
        Update OLTM matrix
        If (next parent node < previous parent node)
            Outside Iteration Without Improvement (OIWI) ← 0
        Else
            Outside Iteration Without Improvement (OIWI) is increased by 1

    } while {both OIWI and EOIL have not exceeded specified numbers)

    Identify the new restart by using the OLTM matrix
    Check the new restart against the previous restarts.
    Next initial solution ← new restart configuration

} while (the number of restart has not exceeded specified number)
```

Terminate the Outside Search
Return the best outside solution (machine location configuration) together with its best inside solution (part machine assignment) as the best solution found so far.

SUBROUTINE-INSIDE SEARCH

Start with the initial Part-Machine Assignment passed by the Outside Search.
Determine the parameters of tabu search used for the Inside Search
Initialize the Inside Tabu List (ITL), Inside Candidate List (ICL) and Inside Index List (IIL)
Evaluate the initial part-machine configuration
ICL ← initial part-machine configuration
IIL ← initial part-machine configuration
Inside Aspiration Value (in_AL) ← Current initial solution
Current parent node ← Initial part-machine configuration
Do
{
    For each seed generated from the current parent node
    {
        The best inside solution ← large number
        If (seed ∈ ICL), Skip it.
        Evaluate the seed
        If (inside move ≠ tabu) or (inside move = tabu but in_AL is satisfied)
                If (seed < the best outside solution)
                {
                    ICL ← current seed
                    ITL ← current move
                    The best inside solution ← current seed
                }
    }
    Set the next parent node with the current best move
    Next parent node ← current best inside solution
    If (next parent node < in_AL), Update in_AL
    If (next parent node = local optima)
    {
        IIL ← current best inside configuration
        Entries into Inside Index List (EIIL) is increased by 1
    }
    If (next parent node < previous parent node)
        Inside Iteration Without Improvement (IIWI) ← 0
    Else
        Inside Iteration Without Improvement (OIWI) is increased by 1

} while {both IIWI and EIIL have not exceeded specified numbers)

Terminate the Inside Search
Return the best inside solution (part-machine assignment) to the outside search