ASSEMBLY LINE BALANCING WITH TASK PARALLELING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZLEM KAPLAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
IINDUSTRIAL ENGINEERING

SEPTEMBER 2004

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Çağlar GÜVEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Meral AZİZOĞLU Co-Supervisor	Assoc. Prof. Dr. Nur Evin ÖZDEMİREL Supervisor

Examining Committee Members

Prof. Dr. Ömer KIRCA	(METU, IE)	_____
Assoc. Prof. Dr. Nur Evin ÖZDEMİREL	(METU, IE)	_____
Prof. Dr. Meral AZİZOĞLU	(METU, IE)	_____
Assoc. Prof. Dr. Levent KANDİLLER	(METU, IE)	_____
Asst. Prof. Dr.Mehmet Rüştü TANER	(Bilkent, IE)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Özlem Kaplan

Signature :

ABSTRACT

ASSEMBLY LINE BALANCING WITH TASK PARALLELING

Kaplan, Özlem
M.S., Department of Industrial Engineering
Supervisor : Assoc. Prof. Dr. Nur Evin Özdemirel
Co-Supervisor: Prof. Dr. Meral Azizoğlu

September 2004, 89 pages

In this study, we consider single model deterministic Assembly Line Balancing problem with task paralleling. The objective is to minimize the total cost which is composed of station opening cost and task dependent equipment cost. A branch and bound algorithm that allows two-level task paralleling is proposed. A heuristic algorithm is also developed both for obtaining efficient upper bounds to branch and bound and for achieving good approximate solutions for large sized problems. Computational experiments are conducted to investigate the effects of experimental parameters on the cost-related and algorithm-related performance measures. The exact algorithm results are compared to the proposed heuristic algorithm results, station paralleling results and optimal solutions without paralleling.

Keywords: Assembly Line Balancing, Task Paralleling, Branch & Bound Algorithm

ÖZ

OPERASYON PARALELLEME İLE MONTAJ HATTI DENGELEME

Kaplan, Özlem
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Nur Evin Özdemirel
Ortak Tez Yôneticisi: Prof. Dr. Meral Azizoğlu

Eylül 2004, 89 sayfa

Bu çalışmada, tek modelli montaj hattının operasyon paralelleme ile dengelenmesi problemi ele alınmıştır. Amacımız toplam istasyon açma maliyeti ve ekipman maliyetinden oluşan toplam maliyetin en aza indirilmesidir. Problemin çözümü için iki seviye operasyon paralellemeye izin veren bir Dal-Sınır algoritması önerilmektedir. Dal-Sınır algoritması için bir üst sınır belirlemek ve büyük ölçütteki problemler için optimale oldukça yaklaşık sonuçlar elde etmek için bir sezgisel algoritma geliştirilmiştir. Deneysel parametrelerin maliyetle ilgili ve algoritmanın performansı ile ilgili ölçütler üzerindeki etkilerinin incelenmesi için deneyler yapılmıştır. Operasyon paralelleme yapan optimal algoritma, sezgisel algoritma, istasyon paralelleme algortiması ve paralelleme yapmayan optimal algoritma ile karşılaştırılmıştır.

Anahtar Kelimeler: Montaj Hattı Dengeleme, Operasyon Paralelleme, Dal-Sınır Algoritması

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Assoc. Prof. Dr. Nur Evin Özdemirel and co-supervisor Prof. Dr. Meral Azizoğlu for their guidance, advice, criticism, encouragements, patience and insight throughout the research.

I would like to give my special thanks to Eray Esengin, my husband, for his technical assistance and patience during my thesis study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE

CHAPTER 1

INTRODUCTION

Assembly line is an arrangement of workers, machines and equipment in which the product to be assembled passes from one station to another until it is completed (Nof et al., 1997).

Nearly all goods of daily life are made by mass production and in mass production a large number of the same product is produced. Division of labor becomes a necessity for large production volumes and assembly lines are the most preferred production systems when high production rates are desired.

In a flow-line production system tasks are assigned to stations according to the technological sequence of operations. A flow-line production system is generally organized as an assembly line because of its serial layout. In an assembly line, products are consecutively moved from station to station along a mechanical material handling equipment, usually a conveyor belt. At each station, a part of the assembly work is completed (Scholl, 1999).

Output rate of an assembly line is determined by the cycle time, which is the duration between two consecutive products. The work content in a workstation is limited by the cycle time. An efficient assembly line system requires nearly even distribution of the total work content among stations. Each task in an assembly line production system is assumed to be indivisible, i.e. a further division of tasks causes additional tasks and so the total work content is increased. While assigning tasks to stations, the precedence relations and some system specific constraints must be satisfied.

Assembly lines are used in different industries and for different product types. The first assembly line is implemented in 1913 by Henry Ford in automotive industry. By this way the cost of a car is reduced by approximately 60% and a considerable amount of time is saved (Nof et al., 1997). Some other noteworthy industries that employ assembly lines are home furnishings, electrical equipment, and computer manufacturing. Since it is necessary to use different production systems for different product types, there is a wide range of assembly line balancing (ALB) problems in the literature. ALB problem assigns the tasks to workstations so that prespecified task precedence restrictions are satisfied and objectives are optimized.

A number of classification schemes, some of which will be discussed here, exists for assembly lines. Assembly line production systems are generally classified according to the types of products assembled, the manner in which the material flows through the line, and the nature of task times. There are three kinds of assembly lines according to the number of products assembled on the line. These are:

1. Single-Model Lines: A single product is manufactured in large volumes. Each station has to perform the same tasks.
2. Mixed-Model Lines: Several different models produced on the same line simultaneously. Since there are only minor differences among production processes, the same assembly line can be used for production.
3. Multi-Model Lines: Different models are produced on the line, however not simultaneously. Due to significant differences in production processes, assembly line is rearranged between product switches.

According to the way the material flows through the line, there are two types of assembly lines: paced and unpaced assembly lines. In a paced assembly line, all stations have the same amount of time to perform the tasks. At the end of this time period, the product to be assembled is passed automatically to the next station. In unpaced lines, a task is transferred to the next workstation once it is completed. So the transfers are not synchronous.

Another classification of ALB is according to the assumption on the nature of operation times: deterministic operation times, stochastic operation times, and dynamic operation times. If the expected task time variability is sufficiently small, then task times are assumed as deterministic. Task time variability is large, in particular with human workers. Even in automated flow lines, due to some disruptions like machine breakdowns and tool breakages, task times can vary considerably. Under these circumstances, stochastic task times are assumed. Dynamic operation times are possible if learning effects or successive improvements in production processes are taken into account. As mentioned by Scholl (1999), in a new assembly line system, task times are reduced after the workers become familiar with work.

There are two versions of the assembly line balancing problem according to the objective function:

- The first type, type I, requires feasible assignment of tasks to stations so that the workload assigned to each station does not exceed a prespecified cycle time and the number of stations is minimized.
- The second type, type II, of the problem fixes the number of stations and requires a feasible assignment of tasks to stations such that the cycle time is minimized.

In these two types of problems, there can be additional objectives such as minimizing the cost of equipment necessary for the tasks.

ALB problem belongs to the NP-hard class of combinatorial optimization problems (McMullen and Tarasewich, 2003). The problem can be formulated as sequencing problem, hence it has a finite but extremely large number of feasible solutions. Without any constraints, there are N! different sequences of N tasks. However, precedence relations or any specific constraints can reduce this number significantly. Erel and Sarin (1998) mention that when the number of precedence relations is r, the possible number of distinct sequences reduces approximately to N! / 2r. ALB problem has been studied extensively for the last 40 years and a lot of research has been done so far in this area.

Another classification is done according to the number of workstations used in each stage. Serial and parallel workstations are two variants. Majority of the ALB studies consider serial station environments. Increased flexibility in assigning tasks, decreased failure sensitivity of the assembly line production system, and shortened cycle times necessitate the use of paralleling in assembly lines. Despite its importance, the literature on assembly line balancing with paralleling is very limited.

In a traditional assembly line, the production rate of the system is limited by the longest task time, i.e., the cycle time cannot be less than the longest task time. Also, a task can only be assigned to a particular station. By allowing paralleling in an assembly line, the production rate is no more limited by the longest task time, and a task can be assigned to more than one station without violating the rule that a task is indivisible. The paralleling in ALB can be one of the two types: paralleling of stations and paralleling of tasks.

If a station is paralleled in an assembly line, production facilities and the total work content of that particular station is duplicated. If two parallel stations are available, the work pieces are alternately fed into one of the stations performing identical set of tasks. By paralleling a station m times, the maximum allowed work content is increased by m times, thereby increasing the maximum achievable production rate. Another advantage of parallel stations is the resulting reduced idle times by fitting tasks to stations more tightly because the work content allowed at a station is increased by the number of times the station is paralleled. Parallel stations also increase the reliability of a production system since, in case of a disruption in one station, the others can continue to work.

Despite various advantages of station paralleling, the additional cost incurred by duplicating all tasks in a station should also be taken into account while designing parallel lines.

The second type of paralleling is task paralleling in which a task is allowed to be performed at more than one station. If a task is to be paralleled twice, it is split up

into two dummy smaller tasks, each having half of the task time and the same precedence relations as the original task. Although this dummy task is assigned to different stations, the operation is performed at one of the stations alternating in each cycle. Hence the indivisibility assumption of tasks still holds when the tasks are paralleled.

By task paralleling, the effective task time is reduced by a factor of the number of times a task is paralleled, and the cycle time is no more restricted with the maximum task time. Furthermore, an assembly line with parallel tasks is more flexible. Despite these advantages, the additional cost incurred by duplicated equipment must be considered.

In this study, an exact algorithm is developed for single model deterministic type-I ALB problem with task paralleling. The objective function is to minimize the total cost which is composed of station opening cost and task dependent equipment cost. We propose a branch and bound algorithm that allows two-level task paralleling. A heuristic algorithm is also developed both for obtaining efficient upper bounds to branch and bound and for achieving good approximate solutions for large sized problems. To the best of our knowledge, our study proposes the first exact solution procedure which considers station opening cost and task dependent equipment cost simultaneously by allowing task paralleling.

The thesis consists of six chapters. In chapter 2, the general terminology used in defining ALB problem is introduced, some performance measures are defined, and the paralleling concept in ALB is explained. Chapter 3 is devoted to the literature survey on the ALB problem with paralleling. In Chapter 4, the mathematical formulation is presented and the details of the proposed heuristic and the branch and bound algorithm are explained. In Chapter 5, the experimental parameters used are introduced and the effects of these parameters on the performance measures are analyzed. Chapter 6 concludes the study and presents the suggestions for the future work.

CHAPTER 2

AN OVERVIEW OF ASSEMBLY LINES WITH PARALLELING

In this chapter, we first give the terminology and performance measures used in ALB and then explain the paralleling concept with some examples.

2.1 TERMINOLOGY USED FOR ASSEMBLY LINES

A task (operation) is the smallest, indivisible work element of the total work content. Task time is the necessary time to perform a task. Task time is sometimes referred to as operation time or processing time.

A station (or workstation) is a location along the assembly line where a subset of tasks is performed. Station load is defined as the set of tasks assigned to a particular station. Station time or work content is the sum of the task times of these tasks. Cycle time is the time between the completion times of the two consecutive end products. Hence, cycle time is the maximum available time in a station to operate on a product. Production rate equals the reciprocal of the cycle time. Precedence relations are the technological restrictions that affect the ordering of the tasks to be performed. Precedence graph (precedence diagram) is the graphical representation of these relations. A precedence diagram contains nodes representing the tasks and arcs between nodes i and j, for all i, j such that task i precedes task j. An example of a precedence graph is given in Figure 2.1.

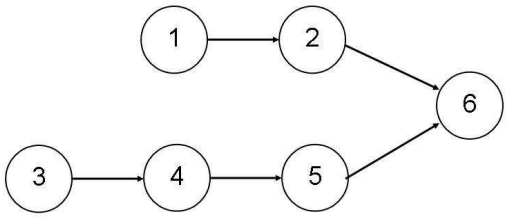

Figure 2.1: Example Precedence Diagram.

The predecessors of a task are the set of all the tasks that must be completed before starting to perform this task. The immediate predecessors of a task are the set of tasks that must be completed just before starting to perform this task. The successors of a task are the set of all the tasks that cannot be performed before completion of this task. The immediate successors of a task are the set of tasks that can be performed just after completion of this task. According to Figure 2.1, all tasks are predecessors of task 6. Task 1 is the immediate predecessor of task 2; tasks 2 and 5 are the immediate predecessors of task 6. Task 6 is the immediate successor of tasks 2 and 5, and successor of all tasks.

Precedence relations can also be represented by a matrix, called precedence matrix. For example precedence matrix for Figure 2.1 can be constructed as in Table 2.1.

Table 2.1: Example Precedence Matrix.

	1	**2**	**3**	**4**	**5**	**6**
1	-	1	0	0	0	0
2		-	0	0	0	1
3			-	1	0	0
4				-	1	0
5					-	1
6						-

There are n rows and n columns for a problem with n tasks. In our example, n is 6. The matrix is symmetric with the upper triangular part having numbers 0 and 1

according to the precedence relations. If task i is an immediate predecessor of task j, then 1 is placed in row i and column j position in the matrix.

Flexibility ratio, FR, is a measure indicating the relative frequency of the precedence relations. It is defined as the number of zero entries in the precedence matrix divided by the total number of entries. The FR value for the example precedence matrix can be found as 20/30, which is equal to 0.667.

Each task requires a specific equipment to be placed in the station where this task is assigned. A task may require different or the same equipment with the other tasks.

2.2 PERFORMANCE MEASURES

The most commonly used performance measures in the assembly lines together with the ones used in this study are as follows:

1. Idle time of a station: This is the difference between predefined cycle time and work content of a workstation, for a paced assembly line. Total idle time is the sum of the idle times over all stations.
2. Balance delay: This is the ratio of total idle time to the total time spent by the product from the beginning to the end of the line (Erel and Sarin, 1998). The total time spent is the number of stations multiplied by cycle time for a paced system.
3. Line efficiency: This is the ratio of the total work content of the line to the total time spent by the product from the beginning to the end of the line.
4. Number of stations: This number measures the optimal number of stations required to balance the line.
5. Total cost: This is the sum of the total cost of station opening and total equipment cost.
6. Number of paralleled tasks: This measure shows the number of paralleled tasks in the optimal solution.

As can be understood from the above definitions, line efficiency can be defined in terms of balance delay as follows: Line Efficiency = 1 – Balance Delay. Therefore, balance delay is inversely proportional to the line efficiency.

For paced lines, balance delay is equivalent to total idle time as well. Total idle time can be considered as a measure of design effectiveness of the line. Balance delay is the inefficiency that results from idle time due to imperfect allocation of work among stations.

If equipment cost is not taken into account in the objective function, total cost is defined as the number of stations multiplied by the cost of opening a station. In our case, we also consider the equipment cost in addition to the station opening cost. We use total cost, the number of stations, the number of equipments used and the number of paralleled tasks as the performance measures in our study.

2.3 PARALLELING OF STATIONS

We illustrate the concept of station paralleling on an example problem taken from Bard (1989). The cycle time to satisfy the demand is 55 minutes. The precedence graph of the example problem is given in Figure 2.2, where numbers on the nodes are the task times. Suppose that each task requires different equipment and the costs of the equipments are identical and equal to 5. In addition we assume that the station opening cost is fixed to 50. The optimal solutions for no station paralleling and station paralleling cases are given in Figures 2.3 and 2.4, respectively. In these figures, numbers in boxes represent the tasks assigned to stations, and figures in parentheses are the station times.

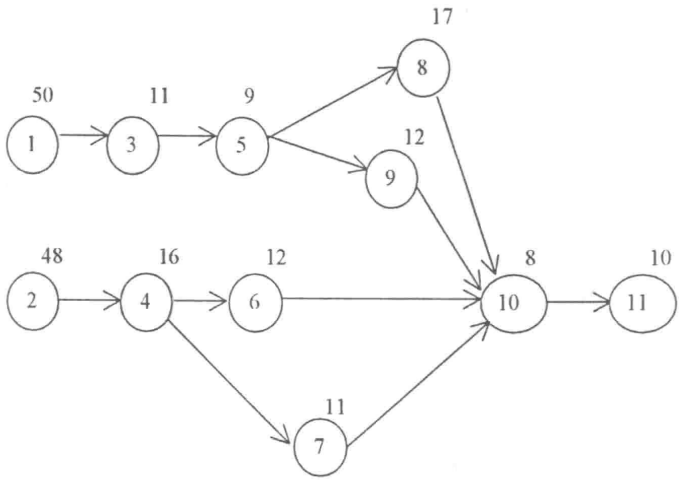

Figure 2.2: Precedence diagram for example problem I.

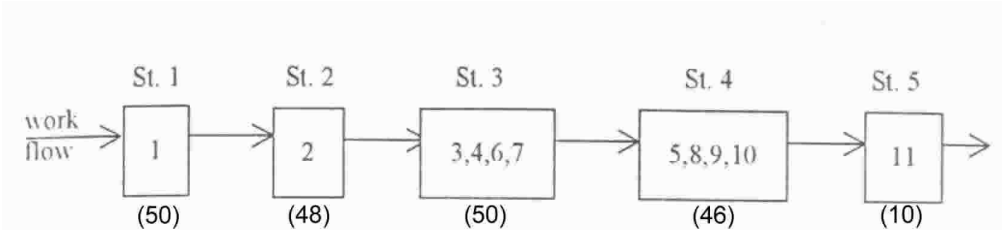

Figure 2.3: Solution to the example problem I without station paralleling.

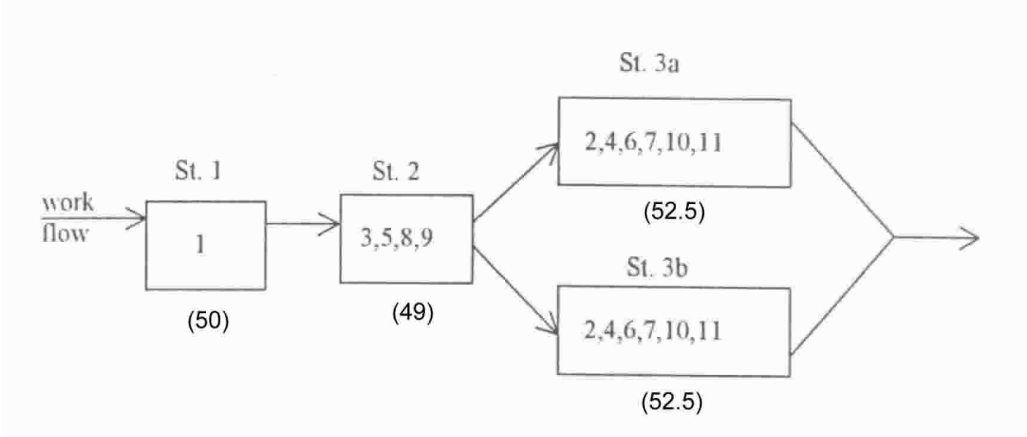

Figure 2.4: Solution to the example problem I with station paralleling.

In the solution of Figure 2.3, there is no station paralleling and in the solution of Figure 2.4 station paralleling is allowed. As can be seen from Figure 2.3, the number of stations required is 5 and the total idle time for this configuration is 71 minutes. Then,

Balance Delay (BD) = 71 / (5 × 55) = 0.26
Efficiency of the line = 1 – 0.26 = 0.74 => Efficiency of the line is 74%
Total Cost = (5 × 50) + (11 × 5) = 305

As can be seen from Figure 2.4, the number of stations required is 4 and the total idle time for this configuration is 16 minutes. Then,

Balance Delay (BD) = 16 / (4 × 55) = 0.07
Efficiency of the line = 1 – 0.07 = 0.93 => Efficiency of the line is 93%
Total Cost = (4 × 50) + (17 × 5) = 285

It is seen that, when station paralleling is allowed, the total number of stations reduces to 4 and a cost saving of 20 is obtained. The line efficiency achieved by station paralleling is 93%. With station paralleling a more efficient line with a lower cost is obtained.

2.4 PARALLELING OF TASKS

We illustrate task paralleling on a simple example problem. The precedence diagram for example problem II is given in Figure 2.5, with task times on the nodes.

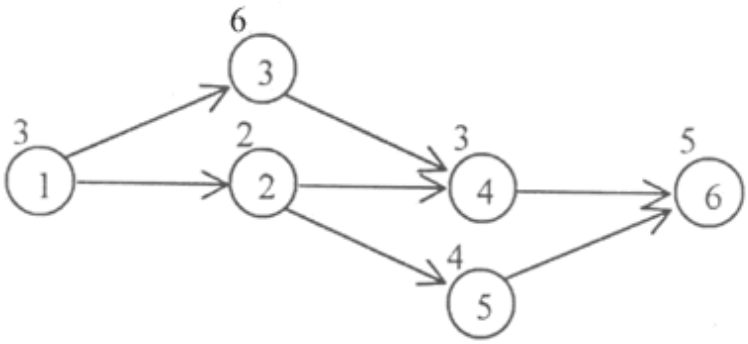

Figure 2.5: Precedence diagram of example problem II.

Suppose, the station opening cost is 50, the equipment cost is 5 for all equipment types and the cycle time is 6. In the example problem, task 3 with task time 6 can be paralleled such that they can be replaced by two new tasks with task time of 3.

The solution with assigned tasks is represented in Figure 2.6. Task 3 is paralleled in stations 1 and 3.

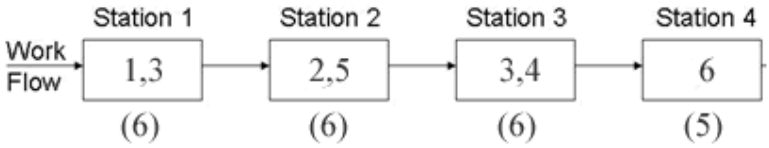

Figure 2.6: Solution to the example problem II with task paralleling.

For the solution with task paralleling:

Balance Delay (BD) = 1 / (4 × 6) = 0.04

Efficiency of the line = 1 – 0.04 = 0.96 => Efficiency of the line is 96%

Total Cost = (4 × 50) + (7 × 5) = 235

For the solution without task paralleling, the required number of stations is 5. The efficiency of the line is calculated as follows:

Balance Delay (BD) = $7 / (5 \times 6) = 0.23$

Efficiency of the line = $1 - 0.23 = 0.77$ => Efficiency of the line is 77%

Total Cost = $(5 \times 50) + (6 \times 5) = 280$

Note that, by task paralleling the efficiency of the line is increased from 77% to 96%. In addition, a cost saving of 45 is obtained with task paralleling since total cost decreases from 280 to 235. Therefore, task paralleling both achieves better line efficiency and lower total cost for example problem II. When total cost is the main concern, task paralleling should be preferred to no paralleling alternative.

CHAPTER 3

LITERATURE ON ASSEMBLY LINE BALANCING WITH PARALLELING

ALB problem solution procedures in the literature are divided into two categories as heuristic methods and exact methods. Exact methods include integer programming, dynamic programming procedures, and branch and bound algorithms.

Among heuristic methods priority ranking approach is commonly used. In this approach, the tasks are ranked according to a given criterion or priority rule and assigned to the stations. Branch and bound based methods like tree search or enumerative methods are also common heuristic approaches that are widely used. Trade and transfer methods interchange the tasks among stations. They start with an initial balance and try to improve it. Random sampling methods assign tasks to stations by selecting a rule from a set of rules randomly (Ghosh and Gragnon, 1989).

In the literature there are many exact and heuristic algorithms developed for the single model deterministic ALB problems. Due to the complex nature of the ALB problems heuristic procedures are more common than the optimum-seeking algorithms. For more realistic ALB problems considering mixed-models and stochastic parameters, the solution procedures published are mostly heuristic (Erel and Sarin, 1998).

In this study, we address the single model deterministic ALB problem with task paralleling that applies to the single-model assembly lines with deterministic times. Since the literature on ALB problems with paralleling is very limited, our literature review includes not only single model deterministic ALB problems, but also mixed model and stochastic ALB problems with station or task paralleling.

Assembly line balancing literature with paralleling is very limited compared to the large number of studies on serial station ALB. Studies on ALB with station and task paralleling that have been reached so far are summarized in Tables A.1 and A.2. In these tables, the main characteristics and assumptions of the problem, the objective function and the solution approach of each study are given. The studies are given in chronological order. The reference information (title) of the study is presented in the first column. The assumptions of the problem are summarized in the second column. The objective function and the solution approach of the problem are reported in the third and fourth columns, respectively. Main conclusions from the computational results and the performance of the proposed methods are discussed in the fifth column of the table. Other system specific characteristics including the nature of paralleling are explained in the last column. These studies are discussed in the following sections.

Majority of the literature on assembly line balancing has concentrated on serial systems. Despite its several important benefits there are only few studies that address parallel workstations. In section 3.1 we review the single model deterministic assembly line balancing.

The benefits that can be achieved through paralleling are improved balance efficiency, shorter cycle times and increased reliability of the line (Bukchin and Rubinovitz, 2003).

3.1 SINGLE MODEL DETERMINISTIC ASSEMBLY LINE BALANCING

Some noteworthy studies on parallel stations are Pinto et al. (1981), Sarker and Shantikumar (1983), Bard (1989), Suer (1998), Ege (2001) and Bukchin and Rubinovitz (2003). Pinto et al. (1975) developed a branch and bound procedure for selecting tasks to be paralleled. The objective is to minimize the total cost including labor cost (both regular cost and overtime cost) and the equipment duplication costs. They state that despite the cost of additional facilities for performing the paralleled tasks, the paralleled line may be a more efficient method for increasing production

compared to alternatives like overtime, a new assembly line, subcontracting and buffer stocks. The objective is expressed in terms of manpower cost. To compare the cost of fixed facilities with labor cost, the fixed costs are converted to equivalent manpower.

According to Pinto et al. (1975), the main problem in balancing assembly lines with task paralleling is to decide which tasks are to be paralleled to minimize the total station opening cost. The proposed branch and bound algorithm proceeds by partitioning the set of all tasks into subsets of partial combinations. A partial combination is made up of tasks that can be classified into three mutually exclusive states: fixed paralleled, fixed not paralleled and undecided. A full combination is achieved by explicitly fixing all tasks as either to be paralleled or not paralleled. At each node the maximum gain from paralleling a task which is the difference between upper and lower bound values of that node is calculated, if the maximum gain from paralleling a task at this node is less than the equivalent fixed cost, then this task is not paralleled and eliminated from further consideration for all branches emanating from this node. The maximum level of paralleling is set to two and the method is tested on problems having 10, 14 and 20 tasks.

In a later work, Pinto et al. (1981) extend their branch and bound algorithm to include two-level station paralleling. This algorithm has many similarities with the previous one like considering labor cost and the cost of additional production facilities when a station is paralleled. The steps of the proposed branch and bound algorithm are the same as in Pinto et al. (1975) except that a partial combination is made up of stations that can be classified as station *fixed paralleled*, station *fixed not paralleled* and station yet *undecided*. A branch and bound method based heuristic algorithm is also presented in this paper.

Sarker and Shantikumar (1983) suggest a general approach that can be applied for both serial and parallel line balancing. The proposed heuristic algorithm can deal with ALB problems having task times shorter or longer than the prespecified cycle time. Their approach is composed of two phases. The first phase requires the assignment of tasks to stations according to the longest task time rule and in the

second phase the balance loss of the line is reduced by trades and transfers. Phase II guarantees a better line balance over Phase I by distributing the idle times of some stages among all stations evenly and station paralleling is implemented in the first phase of the algorithm.

Bard (1989) proposes a dynamic programming algorithm that attempts to meet the required cycle time with minimum number of stations while reducing *dead* time at stations by allowing station paralleling and selecting the minimum cost paralleling configuration. A distinguishing feature of the algorithm is that it takes into account the cost of duplicating both tasks and stations. To the best of our knowledge, this study is the only dynamic programming algorithm with paralleling. The author supposes that the advantage of dynamic programming over integer programming centers on the minimum cost function, which can be redefined without altering the form of the recursive relationship or increasing the dimensions of the problem. Another advantage of the dynamic programming approach stated by the author is that it readily permits the recursive relationship associated with the serial ALB problem to be modified to accommodate the cost of paralleling.

Suer (1998) suggests a simple heuristic for designing parallel assembly lines for high production rates with minimum manpower usages. The proposed heuristic is based on a three-phase methodology that includes balancing the assembly line, determining parallel stations and lastly determining the parallel lines. The decisions involved in this problem include how many operators should be assigned to each station (parallel stations) and also how many assembly lines should be configured. In the first phase, tasks are grouped into stations for various configurations. Then, various alternatives are generated for each manpower level by considering different station configurations. In the last phase, after determining the alternative assembly designs, a simple mathematical model is developed to determine the number of assembly lines and their configuration with the objective of minimizing the total number of operators. He concludes that configurations with fewer stations result in better assembly rates.

Ege (2001) developed a branch and bound algorithm that gives optimal solution to the single model deterministic ALB problem by allowing arbitrary level of station paralleling. The objective is the minimization of total station opening and equipment cost. The most notable feature of his algorithm is that it considers task dependent equipment cost and station paralleling simultaneously. He uses the heuristic algorithm, developed by Askin and Zhou (1997) to find an initial solution to his optimizing algorithm. Ege states that paralleling provides great savings, in particular when flexibility is high and when equipment cost is close to station opening cost.

Bukchin and Rubinovitz (2003) show that the assembly system design problem with parallel stations can be treated as a special case of equipment selection problem. A branch and bound algorithm developed for the equipment selection problem is adopted to solve the ALB problem with station paralleling. In the assembly system, several equipment types as well as a human operator that are capable of performing assembly operations are considered. In the objective function weights that are associated with the system costs are used to obtain different line configurations. Selecting different values for the weight factor helps to achieve different line design objectives while using parallel stations. To favor smaller number of parallel stations, a small penalty cost can be added to the weight factor. By assigning different values for weight factors, different objectives can be obtained.

3.2 MIXED MODEL ASSEMBLY LINE BALANCING

Although the mixed-model ALB problem reflects modern manufacturing environments more realistically, it has received little attention in the literature. When the concept of paralleling is added to the complex mixed-model problem, the number of studies becomes even smaller.

McMullen and Frazier (1997) propose an approach for solving the mixed model ALB problem with stochastic task times when paralleling of stations is permitted. The approach presented in this work is a modification of Gaither's (1996) incremental utilization heuristic. The original heuristic assumes deterministic task

times and can only solve single model ALB problem. The modified heuristic deals with stochastic task times and uses composite task durations to address mixed model production. The heuristic algorithm also uses several different alternative task selection rules for task assignment. Paralleling is implemented by allowing multiple workers to exist within cells.

Askin and Zhou (1997) developed a heuristic algorithm that deals with mixed model deterministic assembly line production with station paralleling. The objective considers task dependent equipment cost besides the fixed cost of operating stations and it trades off between station idle time and equipment cost. By using a threshold value station utilization is also considered explicitly. As in Ege's (2001) study, the proposed algorithm does not restrict the number of times a station can be paralleled and the decision on station paralleling is not only based on task times but also on the comparison of incremental equipment cost.

McMullen and Frazier (1998) propose a simulated annealing approach for ALB problem with multiple objectives allowing station paralleling. In the study, task times are assumed to be stochastic. There are two objectives: the extent to which the desired cycle time is achieved and the total design cost of the production line including the equipment cost and the labor cost. The algorithm can be used both in single model and mixed model ALB. Their experimental results show that the approach gives better solutions in terms of cycle time but average solutions in terms of cost.

Vilarinho and Simaria (2002) present a heuristic approach for the mixed model ALB problem with parallel workstations and zoning constraints. The procedure allows the user to control the process to create parallel workstations. For example, the decision maker can define a limit on the maximum number of parallel stations and the conditions under which a station can be paralleled. The primary goal of the model is to minimize the number of stations in the assembly system and the secondary goal is to balance the workload between and within the stations. This algorithm has some common points with the heuristic algorithm developed by McMullen and Frazier (1998) as both use simulated annealing for mixed model multi-objective ALB

problem with parallel workstations. However, the model developed by Vilarinho and Simaria (2002) includes zoning constraints, assumes deterministic task times and the number of paralleled stations is determined by the decision maker where McMullen and Frazier (1998) allow the replication of a workstation as long as its utilization increases. The heuristic proposed by Vilarinho and Simaria (2002) is composed of two stages. In the first stage, the procedure finds a sub-optimal solution to the primary goal and the second stage deals with the secondary goal.

McMullen and Tarasewich (2003) address the mixed model ALB problem with complicating factors of stochastic task durations and parallel workstations. This study deals with the same problem that was introduced by McMullen and Frazier (1997, 1998). They propose a heuristic approach that uses concepts derived from Ant Colony Optimization. The performance of the heuristic algorithm is compared with other approaches like work center loading approaches and simulated annealing. They conclude that the ant colony based approaches can compete with the other heuristic methods.

CHAPTER 4

PROBLEM FORMULATION AND SOLUTION

In this chapter, we describe our approach to solve single model deterministic ALB problem with task paralleling. In the first section, we give the mathematical formulation of the problem together with its assumptions. The steps of the proposed heuristic algorithm are explained in Section 4.2. Our Branch and Bound approach is described in Section 4.3 together with lower and upper bounds used.

4.1 MATHEMATICAL FORMULATION

We study the single model deterministic ALB problem with the following assumptions:

- A single product is assembled on the line.
- The task times are deterministic.
- The cycle time is known and fixed.
- The precedence restrictions, which provide information about the relative processing order of the tasks, are known and fixed.
- Each task requires one of the several types of equipment, which can be a machine, an instrument or a tool that is required to perform the task.
- Each type of equipment has a specific cost. This cost includes purchasing and operational costs.
- There is a fixed cost (including labor and overhead costs) for opening a workstation.
- Two level task paralleling is allowed.
- Maximum task time cannot exceed twice the cycle time.

- When a task is paralleled the required equipment is placed in every station where the task is to be performed.
- A task can be performed at any workstation provided that the equipment for this task is available in that workstation and the precedence restrictions are satisfied meaning that all paralleled and non-paralleled predecessors should have been assigned.

The notation used throughout the mathematical formulation of the problem is as follows:

Indices:

j : Task number = {1, 2, … N}
l : Type of equipment = {1, 2, … D}
k : Station number = {1, 2, … N + M}

Parameters:

c : Cycle time
L : Fixed cost to open a station
N : Number of tasks
M: Number of tasks whose task time exceeds the cycle time
t_j : Processing time of task *j*
IP : Set of task pairs (*u*,*v*) such that task u must immediately precede task *v*
D : Number of equipment types
l(*j*) : Equipment type required by task *j*
A_l : Amortized unit cost for equipment type *l*

Sets:

SK : Set of tasks to be performed on the assembly line
= {1, 2, …,N}
SB : Set of tasks that are paralleled

ST_k : Set of tasks assigned to the k^{th} station

Decision Variables:

$$x_{jk} = \begin{cases} 1, \text{if task j is assigned to station k} \\ 0, \text{otherwise} \end{cases}$$

$$w_{lk} = \begin{cases} 1, \text{if equipment type l is required at station k} \\ 0, \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1, & \text{if task j is paralleled, i.e., } \sum_{k=1}^{N+M} x_{jk} = 2 \\ 0, & \text{if } \sum_{k=1}^{N+M} x_{jk} = 1 \end{cases}$$

$$z_k = \begin{cases} 1, \text{ if station k is opened} \\ 0, \text{ otherwise} \end{cases}$$

The mathematical formulation of the problem that allows two level task paralleling and considers equipment duplication costs is given below:

$$\text{Minimize } Z = \sum_{k=1}^{N+M} L.z_k + \sum_{k=1}^{N+M} \sum_{l=1}^{D} w_{lk}.A_l \qquad (1)$$

Subject to

$$1 \le \sum_{k=1}^{N+M} x_{jk} \le 2 \quad \forall j \qquad (2)$$

$$x_{jk} \le w_{l(j)k} \quad \forall j,k \qquad (3)$$

$$\sum_{j=1}^{N} y_j \frac{t_j}{2} x_{jk} + \sum_{j=1}^{N} (1 - y_j) t_j x_{jk} \le c \quad \forall k \qquad (4)$$

$$x_{vk} \leq 1 - x_{ur} \quad \forall (u,v) \in IP, k \text{ and } r \geq k+1 \tag{5}$$

$$x_{jk} \leq z_k \quad \forall j,k \tag{6}$$

$$y_j = \sum_{k=1}^{N+M} x_{jk} - 1 \quad \forall j \tag{7}$$

$$x_{jk} \in \{0,1\} \quad \forall j,k \tag{8}$$

$$w_{lk} \in \{0,1\} \quad \forall l,k \tag{9}$$

$$z_k \in \{0,1\} \quad \forall k \tag{10}$$

$$y_j \in \{0,1\} \quad \forall j \tag{11}$$

The objective function (1) is to minimize the total station opening cost and total equipment cost. Constraint set (2) ensures that all tasks are assigned to at least one and at most two workstations. Constraint set (3) guarantees that if task j is assigned to station k then the equipment required by the task is placed in the station. Constraint set (4) ensures that the sum of the processing times of the tasks assigned to one station does not exceed the cycle time. Precedence restrictions are considered in constraint set (5) such that if task v is assigned to station k then all its predecessors should have been assigned to station k or one of the prior stations. For paralleled predecessors of task v, this constraint ensures that they are completed, i.e. both halves are assigned to station k or stations prior to k. Constraint set (6) guarantees that if any task is assigned to a station, then the station is opened. Constraint set (7) defines the number of times task j is paralleled. Constraint sets (8), (9), (10) and (11) ensure that the related variables can only take the values of 0 and 1.

The mathematical formulation of our problem is a complex one since it has nonlinear relations, too many constraints and too many integer variables.

4.2 HEURISTIC ALGORITHM

The solutions obtained from the heuristic algorithm are used as initial upper bounds for the Branch and Bound algorithm. Moreover, the heuristic algorithm provides very good solutions in reasonable running times to the problems having large number of tasks.

The heuristic is composed of two parts: Construction and Improvement. In the first part we obtain a feasible solution whereas in the second part we improve the resulting feasible solution.

4.2.1 CONSTRUCTION OF THE SOLUTION

The flowchart for the construction part of the proposed heuristic algorithm is given in Figure 4.1. As a general principle, when assigning the tasks to the stations, the heuristic algorithm first considers the unassigned tasks, whose equipment is present in the current station. Then, the tasks whose equipment is not present are considered for assignment. When there is no task fitable to the current station, the algorithm tries to parallel tasks only if they satisfy the tradeoff criterion given in the flowchart. The tradeoff criterion compares the penalty cost of station idle time (which is calculated as the multiplication of station opening cost and the ratio of remaining station time to cycle time) and the cost of placing extra equipment. Of the tasks to be paralleled, the heuristic algorithm again assigns these whose equipment is already present in the current station. The heuristic algorithm improves that solution by trying to change the stations of the paralleled tasks. In doing this, it takes into account equipment duplications in stations to reduce the total equipment cost.

Detailed description of the steps of the algorithm is given below.

The tasks are initially ordered in their non-increasing order of positional weights as in Askin and Zhou (1997). Positional weights are calculated by summing the processing time of the task and the processing times of all its successors. All tasks

are put into U, which is the set of unassigned tasks. Station index, k, is set to 1 indicating the first station is opened. X_k (set of equipment assigned to station k), V_k (set of tasks assigned to station k) and Z_k (set of paralleled tasks for station k) are empty. T_k (remaining time for station k) is set to the cycle time since there is no task assigned to the station yet, and μ_k (utilization factor for station k) is set to 0. PS (a binary variable indicating the paralleling state, e.g. 0 for non-paralleling state and 1 for paralleling state) is set to 0.

The heuristic algorithm continues until there is no task left in U, i.e. all tasks are assigned to stations. It first tries to find a fitable task whose equipment is already placed in station k. If there is such a task it is assigned to station k and variables V_k, T_k, μ_k, j (current task index) and U are updated as in the flowchart. X_k is not updated here since the required equipment for the newly placed task is already present in station k. If such a task does not exist, then the algorithm tries to find a fitable task whose equipment is not present in station k. If there is such a task, it checks whether PS is one meaning that the algorithm tries task paralleling. If PS is zero, it assigns that task to station k and variables X_k, V_k, T_k, μ_k, j and U are updated. If PS is one, the algorithm checks the tradeoff between the idle workstation time penalty and equipment cost for the task. This tradeoff approach is similar to the one found in the heuristic algorithm of Askin and Zhou (1997). The penalty cost of station idle time is found by multiplication of station opening cost with the ratio of remaining station time to cycle time.

If the equipment cost is less than the idle workstation time penalty, the task is assigned to station k and variables X_k, V_k, T_k, μ_k, j and U are updated. This means that the task is paralleled and its equipment is duplicated. The algorithm continues until there is no fitable task even when unassigned tasks are paralleled, that is, their task times are halved. Then, it checks whether the utilization factor is 1 or not. If it is 1, the station is closed and k is increased by 1, variables X_k, V_k are set to empty, T_k is set to cycle time and μ_k is set to 0 for the next station.

If the utilization factor is less than 1, it checks whether PS is one. If yes, it multiplies the times of tasks present in U by two and sets PS to zero, closes the station (since

there is no more fitable tasks even when they are paralleled) and k is increased by 1, variables X_k, V_k are set to empty, T_k is set to cycle time and μ_k is set to 0 for the next station. If PS is zero, it halves the times of tasks present in U and sets PS to one.

As the algorithm opens a new station, it first checks whether there are paralleled tasks in the previous station. If there are paralleled tasks, then they are also placed in the newly opened station and variables are updated accordingly.

When U is empty, the algorithm stops since all tasks are placed in a station or more than one workstation if they are paralleled.

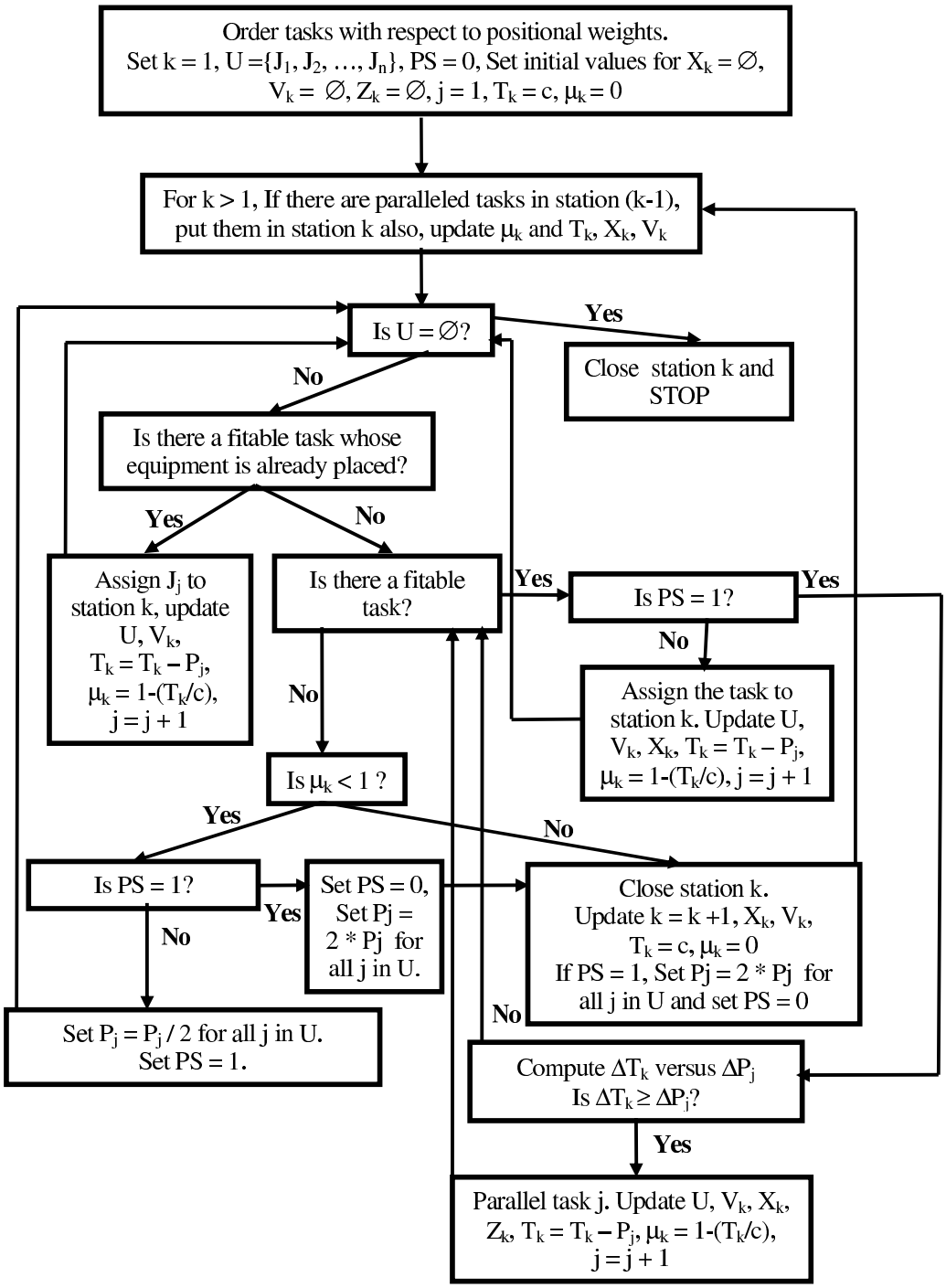

Figure 4.1: The flowchart of the heuristic algorithm.

4.2.2 EXAMPLE SOLUTION CONSTRUCTION

An example problem is solved with the heuristic algorithm. The precedence graph is given in Figure 4.2, where the numbers on the nodes represent task times, and parameters are given as follows:

L (station opening cost) = 50

A_l (equipment cost) = 5

C (cycle time) = 7

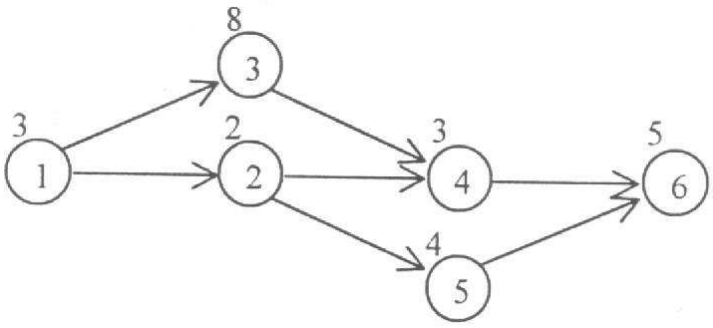

Figure 4.2: Precedence graph for example problem III.

Table 4.1 shows required equipment types and positional weights for each task represented in Figure 4.2.

Table 4.1: Tasks, required equipment types and positional weights for example problem III.

Task (j)	Type of Equipment (l)	Positional Weight
1	1	25
2	2	14
3	3	16
4	4	8
5	1	9
6	5	5

Initialization:

$k = 1, j = 1, \mu_1 = 0, T_1 = c = 7, V_1 = X_1 = Z_1 = \varnothing, U = \{1,2,3,4,5,6\}$

For k = 1:

U is not empty and there is no fitable task to station 1 whose equipment is already placed. Task 1 is fitable to station 1 and PS is zero (i.e. none of the tasks in U are paralleled), so assign task 1 to station 1.
Update $U = \{2,3,4,5,6\}, V_1 = \{1\}, X_1 = \{1\}, T_1 = 7 - 3 = 4, \mu_1 = 1 - (4 / 7) = 0.429, j = 2$

Go back and check if U is empty? It is not
Task 5 uses equipment 1 also, but it is not fitable because of precedence relations. Task 3 has the highest positional weight, but it is not fitable.
Select fitable task 2 from set U, PS is zero (tasks in U are not paralleled), so assign task 2 to station 1.
Update $U = \{3,4,5,6\}, V_1 = \{1,2\}, X_1 = \{1,2\}, T_1 = 4 - 2 = 2, \mu_1 = 1 - (2 / 7) = 0.714, j = 3$

Go back and check if U is empty? It is not.
There is no task fitable to station 1.
Is $\mu_1 < 1$? YES
Is PS = 1? NO
Halve times of tasks in U and set PS to one.

Go back and check if U is empty? It is not.
Task 5 uses the equipment 1 also and is fitable when paralleled. So, assign paralleled task 5 to station 1.
Update $U = \{3,4,6\}, V_1 = \{1,2,5\}, T_1 = 2 - (4 / 2) = 0, \mu_1 = 1 - (0 / 7) = 1, j = 4$

Go back and check if U is empty? It is not.
There is no fitable task to station 1.

Is $\mu_1 < 1$? NO

Close station 1.

Update k = 2, $V_2 = X_2 = Z_2 = \varnothing$, $T_2 = 7$, $\mu_2 = 0$, PS = 0

For k = 2:

Task 5 is paralleled in station 1. So, it is also assigned to station 2.

Update $V_2 = \{5\}$, $T_2 = 7 - (4 / 2) = 5$, $\mu_2 = 1 - (5 / 7) = 0.286$

Check if U is empty? It is not.

There is no fitable task whose equipment is already placed in station 2. There is no fitable task to station 2.

Is $\mu_2 < 1$? YES

Is PS = 1? NO

Halve times of tasks in U and set PS to one.

Go back and check if U is empty? It is not.

Paralleled task 3 is fitable to station 2. Check the tradeoff between cost of paralleling of task 3 (ΔP_3) and station idle time penalty (ΔT_2).

Compute $\Delta P_3 = 5$, $\Delta T_2 = 50 \times (5 / 7) = 35.7$

$\Delta P_3 < \Delta T_2$, then parallel task 3.

Assign paralleled task 3 to station 2.

Update U = {4,6}, $V_2 = \{3,5\}$, $T_2 = 5 - (8 / 2) = 1$, $\mu_2 = 1 - (1 / 7) = 0.857$, j = 5

Go back and check if there is a fitable task to station 2? NO

Is $\mu_2 < 1$? YES

Is PS = 1? YES

Multiply times of tasks in U by two, set PS to zero, and close station 2.

Update k = 3, $V_3 = X_3 = Z_3 = \varnothing$, $T_3 = 7$, $\mu_3 = 0$

For k = 3:

Task 3 is paralleled in station 2. So, it is also assigned to station 3.
Update $V_3 = \{3\}$, $T_3 = 7 - (8 / 2) = 3$, $\mu_3 = 1 - (3 / 7) = 0.571$

Go back and check if U is empty? It is not.
Task 4 is fitable and assigned to station 3.
Update $U = \{6\}$, $V_3 = \{3,4\}$, $T_3 = 3 - 3 = 0$, $\mu_3 = 1 - (0 / 7) = 1$

Go back and check if U is empty? It is not.
There is no fitable task whose equipment is already placed in station 3. There is no fitable task to station 3.
Is $\mu_3 < 1$? NO
Close station 3.
Update $k = 4$, $V_4 = X_4 = Z_4 = \varnothing$, $T_4 = 7$, $\mu_4 = 0$

For k = 4

There are no paralleled tasks in station 3.
U is not empty. There is no fitable task whose equipment is already placed in station 4. Task 6 is fitable and assigned to station 4.
Update $U = \varnothing$, $V_4 = \{6\}$, $T_6 = 7 - 5 = 2$, $\mu_3 = 1 - (2 / 7) = 0.714$

Go back and check if U is empty? It is.
Is PS = 1? NO
Close station 4 and STOP.

Tasks 3 and 5 are paralleled in stations 3 and 1, respectively. Therefore, 7 pieces of equipment are used in 4 stations.

Total Cost = $(4 \times 50) + (7 \times 5) = 235$

The tasks in each station according to the heuristic solution of the problem can be seen in Figure 4.3.

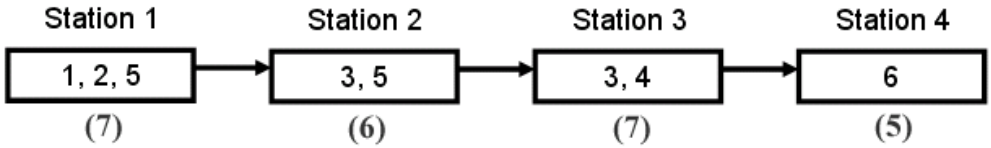

Figure 4.3: Tasks in each station for the example problem solution.

4.2.3 IMPROVEMENT OF THE SOLUTION

After constructing a solution to the problem, an improvement procedure is implemented in order to obtain a better solution by trying to exchange tasks. The improvement procedure only makes exchange of tasks between consecutive stations. The improvement procedure scans all stations one by one and tries to reduce the extra equipment cost incurred by paralleled tasks. It attempts to bring together the two halves of a paralleled task to the same station by exchanging one half of it with a non-paralleled task in the other station.

When exchanging one half of a paralleled task with a non-paralleled task in the other station, the improvement procedure checks if the following rules apply. Firstly, the non-paralleled task must not have common equipment with any other tasks found in its original station, or it must have its equipment in the station where it is going to be assigned. Secondly, the task times of the paralleled and non-paralleled tasks to be exchanged must fit into the remaining times of their new stations. The improvement procedure also checks the precedence relations between tasks before making an exchange.

When exchanging a pair of tasks, the improvement procedure can provide a saving of one or two units equipment costs. If the half of the paralleled task does not share its equipment with another task in the station where it originally resides, one unit

equipment cost is reduced from the total equipment cost. If the non-paralleled task does not share its equipment in the station where it originally resides and it has its equipment in its newly assigned station, another unit equipment cost is reduced.

The improvement procedure prepares two scenarios for exchanging tasks between stations. In one scenario, half of a paralleled task is pulled back from the next station to the previous station and a non-paralleled task is pushed forward to the next station from the previous station. The other scenario pushes half of a paralleled task to the next station from the previous station and pulls the non-paralleled task from the next station to the previous station.

The improvement procedure is effective in particular when the number of tasks is more than 20 and a cycle time is less than or equal to the maximum task time. This is because number of exchange alternatives is increased when total number of stations is increased and when there are more tasks to be assigned to stations.

4.3 BRANCH AND BOUND ALGORITHM

The Branch and Bound algorithm is employed for finding optimal solution to our ALB problem. The algorithm considers two level task paralleling. Some fathoming rules obtained from the literature are used to increase its performance by decreasing the number of nodes. The branching scheme of the algorithm is task-oriented and the algorithm uses best-first search. The flowchart of the developed Branch and Bound algorithm is illustrated in Figure 4.4. Additional parameters and sets, which were not mentioned in the mathematical formulation section and are used in the flowchart of the algorithm, are explained as follows:

Sets:

SE_k : Set of equipment types assigned to station k

SP_k : Set of paralleled tasks in station k

S : Set of tasks in the stack

UT : Set of unassigned tasks (includes second halves of paralleled tasks if they are not assigned yet)

Parameters:

RT_k : Remaining time of station k

BN : Node number at which best solution is found

UB : Upper bound

LB_x : Lower bound of node x

TC_x : Total cost of node x

If the current node is an intermediate node (a partial assignment of tasks), the current station is thought to be still open. If the current node is a leaf node (a full assignment), the current station is thought to be closed. Therefore, for each node of the branch and bound tree the total cost realized so far is calculated as follows:

TC_x = Total station opening cost for the stations closed so far + Total equipment cost for the equipment types placed in the closed stations

Lower bound value for a node is calculated as follows:

$LB_x = TC_x$ + Total station opening cost for minimum number of stations required to be opened for the remaining tasks + Minimum total equipment cost for the remaining tasks

To find the minimum number of stations required to be opened for the remaining tasks (tasks in the current station and all unassigned tasks), the sum of the task times of the remaining tasks is divided by the cycle time. This number is then rounded up to the next integer value and multiplied by the unit station opening cost to find the second term in the lower bound formula. To find the third term, for each type of equipment, the task times of the remaining tasks that use the specified type of equipment are summed, divided by cycle time, and rounded to the next larger integer value. By this way, minimum required number of equipment is found for each equipment type. Multiplying the sum of these numbers with the unit equipment

cost, the last term is obtained. If the current node is a leaf node, its lower bound is equal to its total cost since there are no more tasks to be assigned and all stations are closed.

The algorithm starts with the initialization phase. Sets ST_k, SE_k, SP_k are empty, UT is filled with all the tasks, remaining times for all k are set to cycle time and BN is undefined in the initialization phase. Stack is also empty at the beginning of the algorithm. UB is taken from the heuristic solution of the same problem.

The algorithm searches for independent tasks, which do not have any predecessors. It creates a starting node for each independent task found. If the task time of the independent task is greater than the cycle time, only one starting node, which considers task paralleling, is created for that independent task. If, on the other hand, the task time of the independent task is less than or equal to the cycle time, two starting nodes are created for that independent task, one considering task paralleling and one assigning the independent task to the first station as a non-paralleled task. Then, these starting nodes are added to the stack in increasing order of their lower bound values so that the task with the lowest lower bound stays at the top of the stack.

The algorithm starts each iteration by checking whether the stack is empty or not. This is the termination condition of the algorithm since it stops when the stack is empty meaning that there are no more nodes waiting to be processed. If the stack is not empty, the algorithm removes the node from the top of the stack and starts processing it. When processing a node, the current task under consideration is added to station k, which is either the currently open station or the next station to be opened. Then, values of ST_k, SE_k, SP_k, RT_k, and UT are updated if necessary. SE_k is updated if the equipment required by the task is not already present in station k. SP_k is updated if the task assigned is a paralleled task. UT is updated to indicate if task is completely assigned to station k or if this is the second half of the paralleled task.

Another check is performed after the node is added to the tree. If UT is empty, it means that this node is a leaf of the tree. The algorithm checks if the total cost of the

node is less than the upper bound or not. If yes, this means that the solution found is the best one obtained so far; therefore, BN and UB are updated. Then, the algorithm goes back to check whether the stack is empty or not and continues its cycle by taking the next element from the stack. Since this node is a leaf node, backtracking occurs here. In other words, the algorithm moves upwards in the tree to find a new branch to continue its search for a better solution.

If UT is not empty, it means that this is an intermediate node. Here, the algorithm tries to create a maximum of four subnodes with the following assignment possibilities for the current task under consideration.

- If the task is a full task (not the second half of a previously paralleled task):

1. If the task time is fitable ($t_j < RT_k$) assign it to the current station as a non-paralleled task
2. If half the task time is fitable assign it to the current station as a paralleled task
3. If the task time is less than or equal to the cycle time ($t_j < c$) assign it to the next station as a non-paralleled task
4. If half the task time is less than or equal to the cycle time assign it to the next station as a paralleled task

- If the task is the second half of a previously paralleled task:

1. If half the task time is fitable, assign it to the current station
2. If half the task time is less than or equal to the cycle time, assign it to the next station

These assignment possibilities can be simplified to the four given in the flowchart, which is illustrated in Figure 4.4. The lower bound of the newly created node is no less than the UB. The node is fathomed immediately since it cannot offer a better solution.

There are some fathoming rules applied during the implementation of the Branch and Bound algorithm. The fathoming rules adopted from Scholl (1999) are given below.

1. Do not close a workstation if there is a fitable task without incurring an extra equipment cost: This rule does not consider assigning the task, which has the same equipment type found in the current station, to the next station when creating subnodes.
2. Always branch to the fitable task, which does not incur extra equipment cost, having maximum task time: In the same branching level, if there is a fitable task whose equipment is present in the current station and the task time of this task is greater than or equal to the task times of all other tasks in this branching level, the other branches are fathomed.

In order to avoid the duplication of the same sequence of tasks, which are previously enumerated, in a particular station we always branch to the higher indexed tasks.

After trying to create subnodes, the algorithm checks whether there is any subnode created. If no, it performs backtracking. If yes, the subnodes are added to the stack in increasing order of their lower bound values. Then, the algorithm removes the next node, which offers the lowest lower bound value, from the stack and continues moving downwards in the tree to reach a better solution.

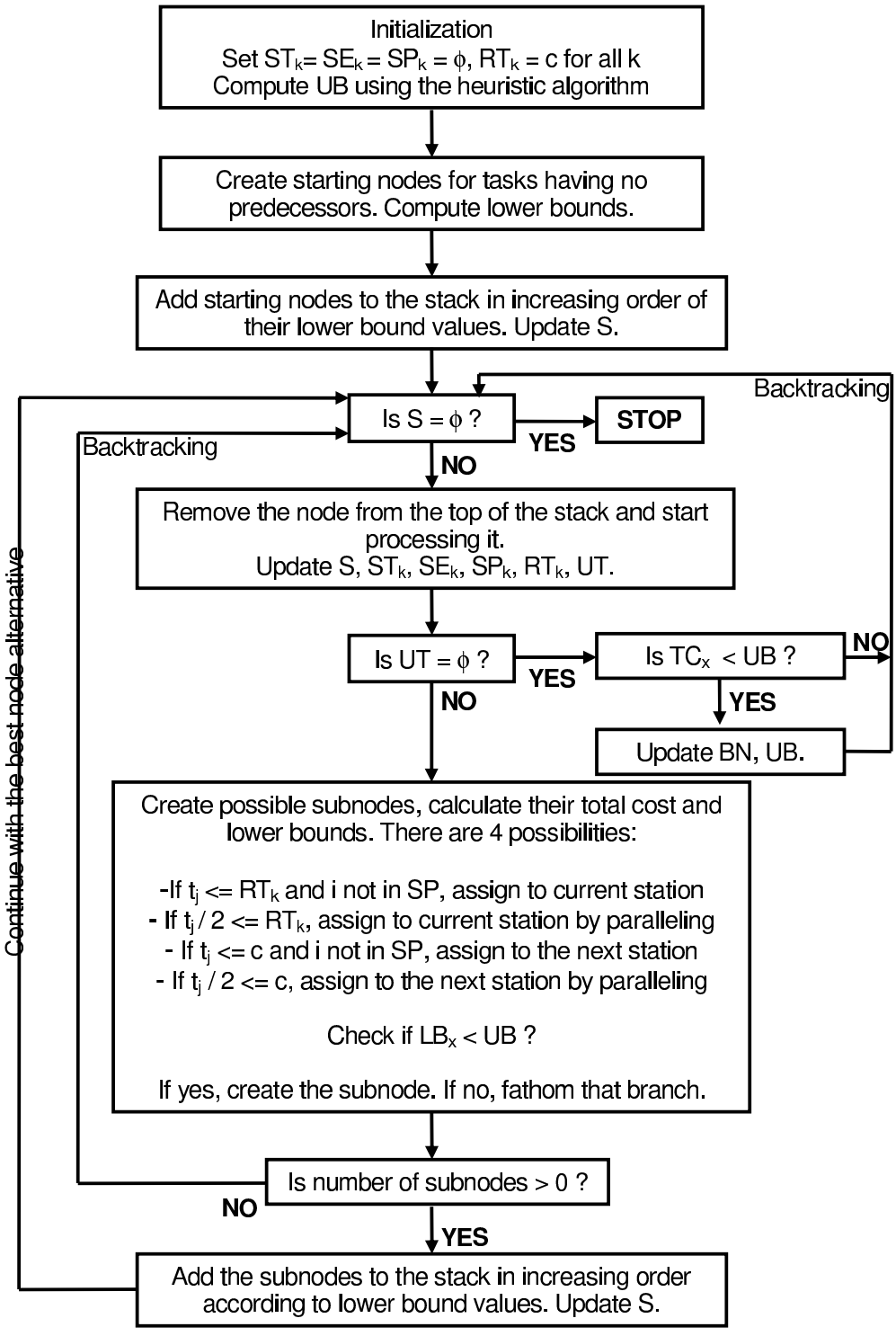

Figure 4.4: Flowchart of the Branch and Bound algorithm.

4.3.1 EXAMPLE PROBLEM SOLUTION WITH BRANCH AND BOUND ALGORITHM

Example problem III given in section 4.2.2 is solved using the Branch and Bound algorithm this time. The precedence relationships among tasks and the task times are illustrated in Figure 4.2. The required equipment types for the tasks are given in Table 4.1. The other parameters for the problem are also the same as before and are restated below.

L (station opening cost) = 50
A_l (equipment cost) = 5
C (cycle time) = 7
Assume UB = 240

The algorithm starts with the initialization phase according to Figure 4.4. In this phase, sets of tasks, equipment types and paralleled tasks for each station are initialized as empty sets. Remaining times are set to the cycle time.

The only independent task in the problem is task 1. Therefore, two starting nodes are created for this task, one considering not paralleling it (called node 1) and one considering paralleling it (called node 2).

Node 2 is fathomed immediately since no task can be placed before task 1 is completed according to the precedence relations. If there had been other independent tasks, these could be placed in the same station as task 1.

The detailed solution of example problem III is given in Appendix B. The remaining of this section illustrates the simplified solution to represent the implementation of the Branch and Bound algorithm.

Remove node 1 from stack;

$ST_1 = \{1\}$, $SE_1 = \{1\}$, $SP_1 = \varnothing$, $RT_1 = 7 - 3 = 4$, $UT = \{2,3,4,5,6\}$

Is $UT = \varnothing$? NO

Possible subnodes are created for node 1:

Node 3: task 2 is assigned to station 1 without paralleling since time of task 2 is less than the remaining time of station 1.

Node 4: task 2 is assigned to station 1 with paralleling since half time of task 2 is less than the remaining time of station 1.

Node 5: task 2 is assigned to station 2 without paralleling since time of task 2 is less than the cycle time.

Node 6: task 2 is assigned to station 2 with paralleling since half time of task 2 is less than the cycle time.

Node 7: task 3 is assigned to station 1 with paralleling since half time of task 3 is less than the remaining time of station 1.

Node 8: task 3 is assigned to station 2 with paralleling since half time of task 3 is less than the cycle time.

Is the number of subnodes > 0? YES

Subnodes 4, 3, 7 are added to the stack in the given order because of their lower bound values and other nodes are fathomed since their lower bound values are greater than the upper bound. Therefore, node 7 stays at the top of the stack for next processing since it has the lowest lower bound value. Figure 4.5 illustrates the situation after subnodes of node 1 are created.

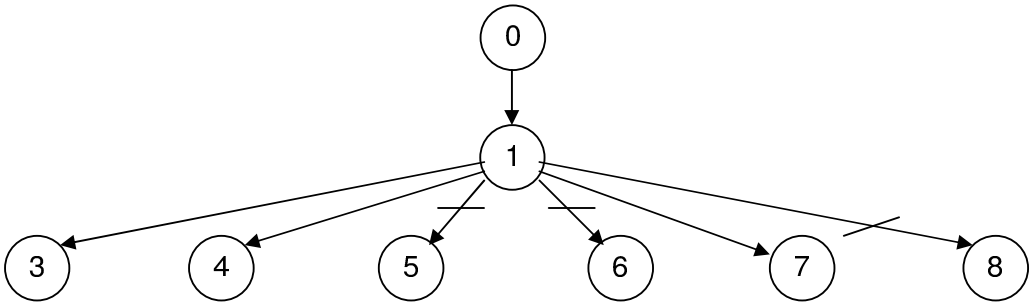

Figure 4.5: Branching after node 1.

Remove node 7 from stack;

$ST_1 = \{1, 3\}$, $SE_1 = \{1, 3\}$, $SP_1 = \{3\}$, $RT_1 = 4 - 4 = 0$, UT = {2,3,4,5,6}

Is UT = ∅? NO

Possible subnodes are created for node 7:

Node 9: task 2 is assigned to station 2 without paralleling.

Node 10: task 2 is assigned to station 2 with paralleling.

Node 11: task 3 is assigned to station 2 with paralleling.

Is the number of subnodes > 0? YES

Subnodes 11, 9 are added to the stack in the given order and subnode 10 is fathomed. Figure 4.6 illustrates the situation after subnodes of node 7 are created.

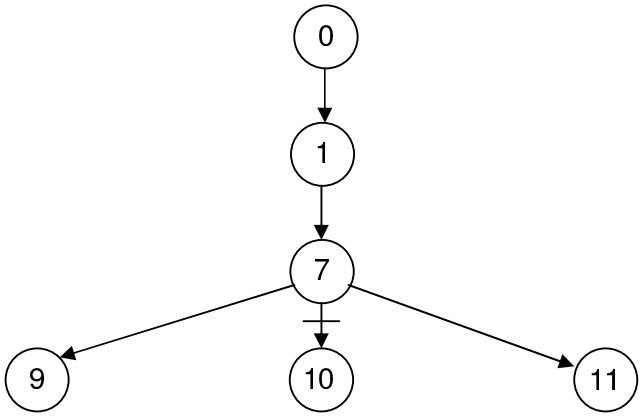

Figure 4.6: Branching after node 7.

Assume a partial schedule with tasks 1 and 2 completely and task 3 is partially assigned to station 1. In station 2, the second half of task 3 and task 4 are assigned and the station is just closed. The following calculations illustrate total cost and lower bound for this condition.

$TC = (2 \times 50) + (5 \times 5) = 125$

$LB = 125 + (2 \times 50) + (2 \times 5) = 235$

The algorithm goes on according to the flowchart illustrated in Figure 4.4 and finds the optimal solution as 235.

CHAPTER 5

COMPUTATIONAL RESULTS

We have conducted experimental runs with our Branch and Bound algorithm, and the proposed heuristic algorithm. We have compared the algorithms with the algorithm proposed by Ege (2001) on the same problem set in order to investigate the effects of station paralleling and task paralleling.

5.1 EXPERIMENTAL PARAMETERS

We generate random test problems using the scheme proposed by Ege (2001). The main characteristics of our randomly generated problems are as follows:

1. **Problem Size (N):** The number of tasks in an assembly network is selected as 10, 15, 20, 25, 30 and 50. The upper limit of 50 is decided such that optimal solution is obtained within three hours.

2. **Distribution of Task Times:** All task times are uniformly distributed between 10 and 100.

3. **Cycle Time (c):** The cycle times are taken as 0.6, 1, 1.4 times the maximum task time. In Ege (2001) for no paralleling option, the cycle times of only 1 and 1.4 times the maximum task time are considered. We hereafter say cycle time as 0.6, implying that the cycle time is 0.6 times the maximum task time.

4. **Flexibility Ratio (FR):** This value varies between 0 and 1. When it is close to 0, the network is not so flexible in terms of precedence relations. As it approaches to 1, the flexibility of the assembly network increases. In our

study, the randomly generated problems having flexibility ratio values of 0.2 and 0.5 are used in experimental runs.

5. **Cost Structure:** Two different cost ratios (L/A_l) are used, namely 30/1 and 30/30. If L/A_l is 30/30, the single equipment cost is equal to the station opening cost. In this case the objective is to optimize the number of stations opened and the number of equipments placed simultaneously. When L/A_l is 30/1, the station opening cost is 30 times as large as a single equipment cost. In this case, the problem is to optimize first the number of stations opened and then the equipment cost, i.e. we have a hierarchy of objectives. Station opening cost is fixed as 30000 for all problems.

6. **Task Equipment Types:** For the randomly generated problems, the number of equipment types is generated from discrete uniform distribution between 1 and N. Once the number of equipment types is set, each task is assigned to an equipment type randomly.

Ten random test problems are generated for each combination of problem size, cycle time, flexibility ratio and cost structure. This gives us a total of 480 problems to be solved with the algorithm proposed by Ege (2001), 720 problems with our heuristic algorithm, 300 problems with our Branch and Bound algorithm.

5.2 PERFORMANCE MEASURES

We use seven performance measures to evaluate the efficiency of our procedures. These performance measures are reported as average and worst case (maximum or minimum of ten problems) for each parameter combination.

1. **Central Processing Unit (CPU) Time:** It indicates the running time of the algorithm on a PC with Pentium-III 450 MHz processor and 256 MB memory. CPU time is given in minutes.

2. **Total Number of Nodes Generated:** The number of nodes evaluated in the Branch and Bound tree.

3. **Total Number of Nodes Generated Until Reaching the Optimal Solution:** The number of nodes evaluated till finding the optimal solution.

4. **Total cost (TC):** The sum of the total cost of station opening and total equipment cost.

5. **Number of stations (NS):** The optimal number of stations required to balance the line.

6. **Number of paralleled tasks (NPT):** The number of paralleled tasks in the optimal solution.

7. **Percentage deviation in terms of total cost:** We have used four different percentage deviation measures to compare total cost values found by
   a. Our Branch and Bound algorithm with task paralleling (TP)
   b. Our heuristic algorithm (H)
   c. Ege's (2001) Branch and Bound algorithm with station paralleling (SP)
   d. Non-paralleling option (NP).

   Note that a, c and d above are all optimal solutions with different paralleling options. The percentage deviations are found as follows

$$H - TP = \frac{H - TP}{TP} \times 100$$

$$SP - TP = \frac{SP - TP}{TP} \times 100$$

$$NP - TP = \frac{NP - TP}{TP} \times 100$$

$$NP - H = \frac{NP - H}{H} \times 100$$

The cost related performance measures are TC, NS, NPT and percentage deviation in terms of total cost whereas algorithm related performance measures are CPU time, total number of nodes generated and total number of nodes generated until reaching the optimal solution.

5.3 DISCUSSION OF THE RESULTS

Our Branch and Bound algorithm and our heuristic algorithm are coded in Visual Basic 6.0 programming language and Ege's (2001) algorithm and the algorithm with no paralleling option are coded in FORTRAN programming language. During the discussion of results, we refer to the algorithm developed by Ege (2001) as the station paralleling algorithm. We also refer to the optimal solution without any paralleling as the non-paralleling option. Since our algorithm only allows two level task paralleling, the station paralleling algorithm is also run by allowing two level paralleling.

5.3.1 EFFECTS OF PROBLEM PARAMETERS ON BRANCH AND BOUND ALGORITHM WITH TASK PARALLELING

The effects of the experimental parameters on the performance measures for our Branch and Bound algorithm are reported in Tables 5.1 and 5.2. In Table 5.1, the cost related measures for Branch and Bound algorithm are summarized. Table 5.2 contains the measures related with the algorithm performance.

Table 5.1: Results related to cost measures for Task Paralleling.

L/A_l	N	FR	c	Total Cost, TC		# of Stations, NS		# of Paralleled Tasks, NPT	
				Avg.	Max.	Avg.	Max.	Avg.	Max.
30/1	10	0.2	0.6	413800	497000	13.4	16	5.7	7
			1.0	219100	281000	7.0	9	1.3	3
			1.4	155600	192000	4.9	6	1.1	3
		0.5	0.6	395500	497000	12.7	16	5.9	8
			1.0	209500	280000	6.9	9	1.7	3
			1.4	154900	191000	4.9	6	0.6	2
	15	0.2	0.6	601100	776000	19.4	20	8.6	12
			1.0	323100	378000	10.3	12	1.6	3
			1.4	226500	257000	7.2	8	2.5	6
		0.5	1.0	306000	378000	9.7	12	2.3	4
			1.4	222800	258000	6.9	8	2.6	5
	20	0.2	0.6	820100	961000	26.4	30	11.6	13
			1.0	428900	507000	13.5	16	4.1	9
			1.4	307300	351000	9.8	11	2.9	4
30/30	10	0.2	0.6	819000	990000	13.5	16	5.4	7
			1.0	480000	600000	7.3	9	0.8	3
			1.4	381000	510000	5.6	8	0.6	2
		0.5	0.6	816000	1020000	12.7	16	5.6	7
			1.0	456000	570000	7.2	9	1.3	3
			1.4	369000	480000	5.5	7	0.5	2
	15	0.2	0.6	1236000	1500000	19.5	21	8.0	10
			1.0	729000	900000	10.9	13	1.2	4
			1.4	600000	720000	8.0	9	1.4	3
		0.5	1.0	693000	840000	10.5	13	1.6	3
			1.4	579000	720000	7.5	9	1.8	4
	20	0.2	0.6	1701000	1860000	26.5	31	10.3	12
			1.0	1017000	1140000	14.6	17	1.8	4
			1.4	816000	900000	10.4	12	1.7	4

As can be seen from Table 5.1, when N increases while the other experimental parameters are kept constant, TC, NS and NPT also increase. With increasing c values, TC, NS and NPT decrease, as increased c values allow more tasks to be assigned to a station. When c is increased from 0.6 to 1, there is a sharper decrease in TC, NS and NPT compared to the case when it is increased from 1 to 1.4. When c changes from 0.6 to 1, the decrease in TC is about 50% whereas when c increases from 1 to 1.4, the decrease in TC is about 30%. This is due to the fact that paralleling becomes inevitable when c is 0.6. Since most of the problems in the set

where FR is 0.5 and c is 0.6 with N = 15 could not be solved they are excluded from the result tables.

Table 5.2: Results related to algorithm performance measures for Task Paralleling.

				CPU Time (min)		# of Nodes		Optimum Node #	
L/A_l	N	FR	c	Avg.	Max.	Avg.	Max.	Avg.	Max.
30/1	10	0.2	0.6	0.0283	0.2488	544.5	3568	180.2	1392
			1.0	0.0034	0.0147	276.9	1074	97.7	334
			1.4	0.0015	0.0035	152.4	323	41.5	102
		0.5	0.6	4.2671	26.6258	4622.8	21050	952.6	3475
			1.0	0.0219	0.1119	919.1	2666	216.4	748
			1.4	0.0211	0.1162	866.4	2810	97.8	206
	15	0.2	0.6	16.3536	101.0456	11556.4	67256	5404.6	29563
			1.0	7.4875	71.5540	6468.4	42982	1882.3	14173
			1.4	0.4999	1.8182	3727.0	9654	550.5	1822
		0.5	1.0	2.8331	13.9926	13037.4	31509	2297.2	8587
			1.4	1.1326	9.0307	7723.0	25075	1272.3	4725
	20	0.2	0.6	155.8344	178.9360	36242.3	96214	3285.0	12508
			1.0	118.2762	166.4718	24960.8	75923	5907.0	18665
			1.4	36.6943	156.7245	10531.9	54656	1280.3	8858
30/30	10	0.2	0.6	0.0142	0.1453	950.3	6954	190.0	2546
			1.0	0.0130	0.1147	485.9	3319	144.4	1030
			1.4	0.0030	0.0082	304.0	740	66.9	150
		0.5	0.6	0.6325	2.6591	4605.7	13371	684.3	2663
			1.0	0.0323	0.2028	1321.8	5251	330.3	1323
			1.4	0.0122	0.0350	899.9	1901	130.5	284
	15	0.2	0.6	52.1256	175.4587	32564.8	104254	6548.2	39654
			1.0	36.1226	145.7583	18256.1	81462	3232.4	22305
			1.4	9.3951	45.5260	14031.8	43975	3140.4	11332
		0.5	1.0	12.6193	95.6510	19562.1	58391	2910.6	7982
			1.4	4.5799	20.6222	17069.6	45896	2466.0	6786
	20	0.2	0.6	102.3236	176.2563	56489.2	132546	9564.3	25654
			1.0	67.4241	142.1564	34505.3	86355	5123.5	13932
			1.4	16.9238	41.8911	15507.9	33647	972.3	4331

If we increase the FR from 0.2 to 0.5 while the other experimental parameters are kept constant, TC, NS and NPT decrease since there are more assignment options due to the more flexible precedence structure.

When the cost structure is switched from 30/1 to 30/30 while the other experimental parameters are kept constant, TC and NS increase whereas NPT decreases. This is

because when the equipment cost is equal to the station opening cost for L/A_1 = 30/30, the algorithm does not favor task paralleling, which would cause an increase in equipment cost.

As can be observed from Table 5.2, as N increases while the other experimental parameters are kept constant, the CPU time, the total number of nodes in the solution and the optimal node number values increase. When FR increases, the CPU time, the total number of nodes in the solution and the optimal node number increase.

When c is increased while the other experimental parameters are kept constant, the CPU time, the total number of nodes in the solution and the optimal node number decrease. When c is increased from 0.6 to 1, there is a sharper decrease in the algorithm related measures compared to the case when it is increased from 1 to 1.4. Again, this results from the paralleling obligation with c of 0.6.

When the cost ratio is switched from 30/1 to 30/30 while the other experimental parameters are kept constant, the total number of nodes in the solution and the optimal node number increase.

5.3.2 EFFECTS OF LOWER BOUNDS AND FATHOMING RULES ON THE BRANCH AND BOUND PERFORMANCE

In order to determine the effectiveness of our lower bound estimated at the root node, we have conducted pilot runs for N = 15 and calculated the percentage deviations of initial lower bound values from the optimal results. The deviations for the test runs are listed in Table 5.3.

Table 5.3: Deviations of lower bound at the root node from optimal solution.

N	FR	c	30/1 % Deviation			30/30 % Deviation		
			Avg	Max	Min	Avg	Max	Min
15	0.2	0.6	30.02	41.69	16.54	27.07	40.00	11.76
		1.0	11.48	14.06	9.88	19.43	35.29	9.09
		1.4	9.48	19.15	1.07	25.03	42.86	14.29
	0.5	1.0	5.22	11.58	1.06	13.40	23.81	4.55
		1.4	7.74	18.09	0.53	20.66	35.71	7.69

The average percentage deviations for FR is equal to 0.2 and 0.5 vary between 9.48 - 30.02 and 5.22 - 7.74, respectively when the cost ratio is 30/1. When the cost ratio is 30/30, the average percentage deviations are between 19.43 - 27.07 and 13.40 - 20.66, for FR is equal to 0.2 and 0.5 respectively. The maximum percentage deviations are 41.69 and 42.86 for cost structures of 30/1 and 30/30, respectively.

In order to detect the importance of lower bound in eliminating partial schedule, we have conducted test runs with the problems when N is 15. The optimal solutions could not be reached within three hours when the flexibility ratio is 0.5. When the flexibility ratio is 0.2 and cost ratio is 30/1, the incorporation of lower bound produces on average about 8 times, 37 times, 31 times less nodes in less CPU time for cycle times of 0.6, 1 and 1.4, respectively over the four problems out of ten that could be solved within three hours. When the cost structure is 30/30, the lower bound implementation performs on the average about 4 times, 37 times, 30 times better in terms of total number of nodes and the CPU time for cycle times of 0.6, 1 and 1.4, respectively over the four out of ten problems that could be solved within three hours.

As can be seen from the above results, the incorporation of lower bound brings considerably high reduction to the total number of nodes in the solution and the CPU time.

For analyzing the effect of fathoming rules, some test runs are conducted with problems of 15 tasks. The algorithm related performance measures are obtained for

the cases with and without fathoming rules. The results are listed in Tables 5.4 and 5.5.

Tables 5.4 and 5.5 show that the total number of nodes in the solution and the CPU time increases when no fathoming rule is applied. The increase in the total number of nodes in the solution reaches upto 4 times of the total number of nodes in the solution when fathoming rules are applied according to the test runs.

Table 5.4: Comparison of fathoming versus no fathoming when cost structure is 30/1.

			Task Paralleling				No Fathoming			
			CPU Time (min)		# of Nodes		CPU Time (min)		# of Nodes	
N	FR	c	Avg	Max	Avg	Max	Avg	Max	Avg	Max
15	0.2	0.6	16.3536	101.0456	11556.4	67256	32.9077	167.5424	18453.8	87159
		1	7.4875	71.5540	6468.4	42982	12.3564	132.2456	10488.0	49836
		1.4	0.4999	1.8182	3727.0	9654	0.9514	3.4560	5768.9	22156
	0.5	1	2.8331	13.9926	13037.4	31509	5.8561	32.3639	31550.5	68375
		1.4	1.1326	9.0307	7723.0	25075	1.9867	8.4822	13206.3	26329

Table 5.5: Comparison of fathoming versus no fathoming when cost structure is 30/30.

			Task Paralleling				No Fathoming			
			CPU Time (min)		# of Nodes		CPU Time (min)		# of Nodes	
N	FR	c	Avg	Max	Avg	Max	Avg	Max	Avg	Max
15	0.2	0.6	52.1256	175.4587	32564.8	104254	78.2564	180.0000	55642.2	107562
		1	36.1226	145.7583	18256.1	81462	89.3330	180.0000	29528.0	104256
		1.4	9.3951	45.5260	14031.8	43975	35.8494	101.4335	47567.8	98943
	0.5	1	12.6193	95.6510	19562.1	58391	34.8579	180.0000	49184.0	103567
		1.4	4.5799	20.6222	17069.6	45896	4.6784	25.6523	18554.0	55623

5.3.3 BRANCH AND BOUND PERFORMANCES FOR TASK PARALLELING, STATION PARALLELING AND NON-PARALLELING

Tables 5.6 and 5.7 illustrate the comparative values for CPU time and number of nodes of our Branch and Bound algorithm, station paralleling algorithm and non-paralleling option for L/A_1 ratios of 30/1 and 30/30, respectively.

It can be concluded from these tables that the number of nodes in the solution and the CPU time for our Branch and Bound algorithm are slightly higher than the ones for station paralleling algorithm. On the other hand, the number of nodes and the CPU times for paralleling algorithms (our Branch and Bound and station paralleling algorithm) are considerably higher than the ones for non-paralleling option. These results hold for both L/A_1 ratios.

Our Branch and Bound algorithm could not find the optimal solution within 3 hours when N is greater than 20 for flexibility ratio of 0.2, and when N is greater than 15 for flexibility ratio of 0.5. Although the number of nodes in our Branch and Bound algorithm and in station paralleling algorithm are close, our algorithm performs more involved operations on one node than the station paralleling algorithm does. Also algorithms are coded in different programming languages. These are the main reasons lying behind the inability of our algorithm to solve optimally the problems having larger number of tasks.

Table 5.6: The CPU time and number of nodes in the solution for $L/A_l = 30/1$.

			Non-paralleled				Station Paralleling				Task Paralleling			
			CPU Time (min)		# of Nodes		CPU Time (min)		# of Nodes		CPU Time (min)		# of Nodes	
N	FR	C	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
10	0.2	1.0	0.0000	0.0000	20.7	39	0.0	0.0000	127.8	268	0.0034	0.0147	276.9	1074
		1.4	0.0000	0.0000	36.6	103	0.0000	0.0000	74.7	148	0.0015	0.0035	152.4	323
	0.5	1.0	0.0001	0.0008	135.8	458	0.0000	0.0000	1529.8	6274	0.0219	0.1119	919.1	2666
		1.4	0.0000	0.0000	200.0	836	0.0002	0.0008	525.7	1668	0.0211	0.1162	866.4	2810
15	0.2	1.0	0.2000	1.0000	161.7	608	0.0002	0.0007	2416.6	9849	7.4875	71.5540	6468.4	42982
		1.4	2.2000	5.0000	404.0	1087	0.5000	2.0000	1124.5	3615	0.4999	1.8182	3727.0	9654
	0.5	1.0	0.0002	0.0008	659.7	1017	0.0004	0.0012	5594.5	14107	2.8331	13.9926	13037.4	31509
		1.4	0.0003	0.0011	1837.1	8733	0.0011	0.0022	4954.4	12582	1.1326	9.0307	7723.0	25075
20	0.2	1.0	0.0008	0.0055	1192.7	9162	0.0023	0.0108	25035.0	109917	118.2762	166.4718	24960.8	75923
		1.4	0.0006	0.0025	1460.5	498	0.0007	0.0017	4125.0	11637	36.6943	156.7245	10531.9	54656
	0.5	1.0	0.0019	0.0078	7428.4	40914	0.0171	0.1275	185830.6	1342771				
		1.4	0.0016	0.0036	6038.1	19492	0.0035	0.0200	27591.1	158965				
25	0.2	1.0	0.0005	0.0027	2031.8	7872	0.0096	0.0495	78938.4	411897				
		1.4	0.0011	0.0039	4400.9	13164	0.0020	0.0075	9284.9	29356				
	0.5	1.0	0.0308	0.2315	181645.4	1326116	0.0959	0.7903	1072081.0	8942624				
		1.4	0.0165	0.0917	109047.1	566043	0.0151	0.0369	110441.9	353227				
30	0.2	1.0	0.0024	0.0114	6219.3	28287	0.0881	0.2582	629021.8	1801597				
		1.4	0.0089	0.0200	18662.1	38886	0.0061	0.0145	46933.8	192186				
	0.5	1.0	0.1075	0.3020	749515.1	3573665	6.7869	54.9928	60876842.0	497235262				
		1.4	0.2822	1.6752	872853.6	4988313	0.2097	0.5378	1482695.0	4123987				
50	0.2	1.0	0.1844	0.3758	361383.7	809255	35.3190	117.9725	191748402.0	620968768				
		1.4	3.1451	11.5481	8734423.0	3545953	1.3564	5.0986	3764499.0	15890523				

Table 5.7: The CPU time and number of nodes in the solution for $L/A_l = 30/30$.

			Non-paralleled				Station Paralleling				Task Paralleling			
			CPU Time (min)		# of Nodes		CPU Time (min)		# of Nodes		CPU Time (min)		# of Nodes	
N	FR	c	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
10	0.2	1.0	0.0000	0.0000	21.6	39	0.0000	0.0002	164.0	415	0.0130	0.1147	485.9	3319
		1.4	0.0000	0.0000	37.8	107	0.1000	1.0000	107.8	206	0.0030	0.0082	304.0	740
	0.5	1.0	0.0002	0.0008	193.6	619	0.0001	0.0005	1562.8	8597	0.0323	0.2028	1321.8	5251
		1.4	0.0000	0.0000	201.1	866	0.0003	0.0008	702.4	2378	0.0122	0.0350	899.9	1901
15	0.2	1.0	0.0003	0.0014	178.9	608	0.0003	0.0017	3454.2	20041	36.1226	145.7583	12354.8	81462
		1.4	0.0012	0.0017	538.4	1115	0.0013	0.0108	2064.4	8630	9.3951	45.5260	14031.8	43975
	0.5	1.0	0.0003	0.0011	1136.9	2560	0.0009	0.0033	13247.6	43935	12.6193	95.6510	16286.1	58391
		1.4	0.0006	0.0019	2169.4	7771	0.0015	0.0058	9151.5	36348	4.5799	20.6222	17069.6	45896
20	0.2	1.0	0.0005	0.0031	1282.2	8994	0.0050	0.0192	54398.0	229593	67.4241	142.1564	34505.3	86355
		1.4	0.0007	0.0028	2047.8	10251	0.0024	0.0097	22781.5	136750	16.9238	41.8911	15507.9	33647
	0.5	1.0	0.0022	0.0089	9964.9	48820	0.0598	0.4350	714819.1	5265827				
		1.4	0.0018	0.0028	8091.2	15920	0.0159	0.0533	185105.9	648355				
25	0.2	1.0	0.0012	0.0090	3152.2	17797	0.0057	0.0308	55411.1	282748				
		1.4	0.0027	0.0094	12262.9	44040	0.0061	0.0233	68204.9	210421				
	0.5	1.0	0.0465	0.2800	282140.9	1656740	0.4699	3.9048	6187955.0	52537967				
		1.4	0.0370	0.2200	252920.3	1385397	0.0702	0.3338	715791.1	3148995				
30	0.2	1.0	0.0031	0.0072	9078.7	37723	0.2422	0.6165	1843920.0	4565624				
		1.4	0.6914	6.5000	109549.2	218802	0.4274	1.8631	1845738.0	7519709				
	0.5	1.0	0.8335	2.3773	1571763.0	3219577	21.9512	93.8548	192835024.0	741974801				
		1.4	3.5709	27.8823	7311064.0	41955276	54.1522	180.0248	127258974.0	418500001				
50	0.2	1.0	0.2992	0.6569	749362.0	1642351	87.5255	180.0882	492932143.0	1088000000				
		1.4	24.7580	82.2419	75592610.0	258513617	123.8396	180.1672	478062578.0	895000001				

5.3.4 EFFECTS OF PROBLEM PARAMETERS ON HEURISTIC ALGORITHM

As can be seen from Tables 5.8 and 5.9, when N increases while the other experimental parameters are kept constant, TC, NS and NPT also increase. With increasing c values, TC, NS and NPT decrease. When c gets larger, the station capacity for accepting tasks increases; therefore, NS and TC (since it depends on NS) decrease. When c is increased from 0.6 to 1, the decrease in TC, NS and NPT is more pronounced compared to the case when it is increased from 1 to 1.4.

If FR increases from 0.2 to 0.5 while the other experimental parameters are kept constant, for c of 1 and 1.4, all cost related performance measures decrease since there are more assignment options due to more flexible precedence structure. However, for c of 0.6, a slight increase in TC is observed when FR increases from 0.2 to 0.5.

In Table 5.9, when the cost ratio (L/A_l) is 30/30, and c is increased from 0.6 to 1, there is a dramatic decrease in NPT. On the other hand, if c is increased from 1 to 1.4, there is a slight increase in NPT in some cases. In this case, stations can hold more tasks and therefore the possibility of using common equipment increases, hence the heuristic favors task paralleling to opening a new station.

When the cost ratio is switched from 30/1 to 30/30 keeping the other experimental parameters constant, TC and NS increase whereas NPT decreases. This is due to the fact that when equipment cost is equal to station opening cost for L/A_l = 30/30, the heuristic algorithm does not favor task paralleling.

The CPU times used by the heuristic are negligible and therefore are not reported.

Table 5.8: Heuristic algorithm results related to cost for $L/A_1 = 30/1$.

N	FR	c	Total Cost, TC		# of Stations, NS		# of Paralleled Tasks, NPT	
			Avg.	Max.	Avg.	Max.	Avg.	Max.
10	0.2	0.6	417200	497000	13.4	16	6.1	7
		1	226200	284000	7.2	9	3.3	6
		1.4	161400	192000	5.1	6	2.3	3
	0.5	0.6	411000	497000	13.2	16	6.0	8
		1	222600	283000	7.1	9	2.3	5
		1.4	161500	192000	5.1	6	1.7	3
15	0.2	0.6	610800	776000	19.6	25	9.3	11
		1	328200	380000	10.4	12	3.6	7
		1.4	236600	289000	7.4	9	3.0	5
	0.5	0.6	614100	775000	19.7	25	8.7	11
		1	319400	383000	10.1	12	3.6	8
		1.4	230200	258000	7.2	8	2.6	5
20	0.2	0.6	826900	962000	26.5	31	13.1	16
		1	443100	536000	14.0	17	5.6	8
		1.4	314000	353000	9.8	11	4.2	7
	0.5	0.6	827100	962000	26.5	31	12.7	16
		1	432800	536000	13.7	17	4.6	8
		1.4	306500	351000	9.6	11	3.5	5
25	0.2	0.6	1008800	1181000	32.3	38	15.7	17
		1	543200	603000	17.1	19	6.9	10
		1.4	383600	418000	11.9	13	4.0	6
	0.5	0.6	1011500	1183000	32.4	38	15.7	19
		1	536300	602000	16.9	19	6.7	10
		1.4	379600	417000	11.8	13	4.3	6
30	0.2	0.6	1217400	1400000	39.0	45	18.9	21
		1	658700	724000	20.8	23	7.5	12
		1.4	463900	516000	14.4	16	5.5	7
	0.5	0.6	1217800	1373000	39.0	44	19.0	23
		1	645200	696000	20.4	22	6.4	11
		1.4	454800	515000	14.1	16	7.6	12
50	0.2	0.6	2019200	2243000	64.6	72	32.0	36
		1	1078800	1200000	33.9	38	12.1	20
		1.4	763000	807000	23.5	25	9.4	13
	0.5	0.6	2017700	2272000	64.6	73	30.2	34
		1	1066400	1162000	33.5	37	12.6	24
		1.4	754900	803000	23.3	25	9.7	14

Table 5.9: Heuristic algorithm results related to cost for $L/A_l = 30/30$.

N	FR	c	Total Cost, TC		# of Stations, NS		# of Paralleled Tasks, NPT	
			Avg.	Max.	Avg.	Max.	Avg.	Max.
10	0.2	0.6	837000	990000	13.6	16	5.2	7
		1	507000	600000	7.9	10	0.9	5
		1.4	402000	510000	5.5	7	1.5	3
	0.5	0.6	843000	1020000	13.8	17	5.2	7
		1	483000	570000	7.4	9	0.9	3
		1.4	390000	510000	5.6	7	0.9	2
15	0.2	0.6	1269000	1500000	20.6	25	7.9	10
		1	738000	900000	11.2	15	0.8	4
		1.4	636000	720000	8.4	10	1.3	3
	0.5	0.6	1260000	1500000	20.3	25	7.3	10
		1	750000	870000	11.0	14	0.9	4
		1.4	633000	720000	7.9	9	1.1	3
20	0.2	0.6	1725000	1890000	27.8	31	10.7	14
		1	1020000	1140000	15.4	18	1.0	4
		1.4	852000	900000	10.7	12	1.9	5
	0.5	0.6	1728000	1860000	27.9	31	10.5	14
		1	987000	1140000	14.4	18	0.9	3
		1.4	819000	900000	10.2	12	1.3	3
25	0.2	0.6	2136000	2370000	34.2	38	12.8	17
		1	1296000	1410000	19.1	22	0.7	6
		1.4	1113000	1200000	13.5	16	1.1	3
	0.5	0.6	2127000	2370000	34.1	38	12.9	17
		1	1278000	1440000	18.9	23	0.3	2
		1.4	1080000	1200000	13.1	16	1.0	2
30	0.2	0.6	2535000	2850000	40.8	45	15.0	20
		1	1536000	1620000	22.8	26	0.8	3
		1.4	1326000	1500000	16.6	20	1.5	4
	0.5	0.6	2553000	2880000	40.9	46	14.9	20
		1	1503000	1590000	22.0	24	0.8	4
		1.4	1284000	1440000	15.8	18	1.0	3
50	0.2	0.6	4281000	4560000	68.6	74	24.8	31
		1	2604000	2640000	37.7	39	0.2	2
		1.4	2286000	2340000	27.4	31	1.0	3
	0.5	0.6	4284000	4620000	68.8	76	24.5	30
		1	2580000	2640000	36.7	39	0.5	2
		1.4	2232000	2310000	26.6	31	0.9	4

For determining the savings of heuristic improvement, test runs are conducted on the problems with N = 20 and FR = 0.2. For the cost structure of 30/1 and when the cycle time is 1, heuristic improvement provides one equipment cost benefit in two problems out of ten. When the cycle time is 0.6, it saves two equipments in only one

problem out of ten. For the cost structure of 30/30 when the cycle time is equal to 1, heuristic improvement brings about one equipment cost benefit in one problem out of ten. Hence, it can be concluded from the test runs that heuristic improvement provides savings upto 0.23% when the cost structure is 30/1 and upto 2.86% when the cost structure is 30/30. The higher saving percentage in cost structure of 30/30 is due to the high equipment cost. Since the number of stations and the number of paralleled tasks is lower when the cycle time is 1.4, there is no improvement observed when N = 20 and FR = 0.2.

5.3.5 COMPARISON OF ALGORITHMS IN TERMS OF OBJECTIVE

The percentage deviations in terms of total cost between the solutions of the algorithms discussed are given in Tables 5.10, 5.11 and 5.12.

Table 5.10 gives the percentage deviations of our heuristic algorithm from our Branch and Bound algorithm in terms of total cost (H - TP). Note that for FR values of 0.2 and 0.5 the average percentage deviations vary between 0.75 – 4.04 and 3.03 – 5.53, respectively when L/A_l is 30/1. When L/A_l is 30/30, the percentage deviations are generally higher. The average percentage deviations for FR values of 0.2 and 0.5 vary between 0.30 – 6.06 and 3.39 – 8.93, respectively when L/A_l is 30/30. These results indicate that our Branch and Bound algorithm provides more savings when the equipment cost is equal to the station opening cost and the flexibility ratio is higher. Since the average percentage deviations are not so high when N is large, the heuristic algorithm can be an attractive alternative to the optimal task paralleling algorithm. Note that the maximum percentage deviation is 18.35 when L/A_l is 30/1 and 30.30 when L/A_l is 30/30.

Table 5.11 reports percentage deviations of our Branch and Bound algorithm compared to non-paralleling option (NP - TP) and station paralleling algorithm (SP - TP) in terms of total cost. Firstly, we compare non-paralleling option with our Branch and Bound algorithm. For flexibility ratios of 0.2 and 0.5 the average percentage deviations fluctuate between 10.13 – 16.05 and 8.39 – 14.88,

respectively when L/A_l is 30/1. The average percentage deviations vary between 1.38 – 5.16 and 3.80 – 5.18, respectively when L/A_l is 30/30. These findings indicate that task paralleling with our Branch and Bound algorithm provides more savings when the equipment cost is much lower than the station opening cost. The maximum savings are 26.04% when L/A_l is 30/1 and 25.00% when L/A_l is 30/30.

Table 5.10: Total cost of heuristic and Branch and Bound task paralleling algorithms.

			% Deviations			
			L/A_l = 30/1		L/A_l = 30/30	
N	FR	c	Avg	Max	Avg	Max
10	0.2	0.6	0.75	6.65	2.35	8.33
		1.0	3.00	13.57	4.95	15.00
		1.4	3.37	16.46	4.73	12.50
	0.5	0.6	4.09	16.53	3.39	11.54
		1.0	5.53	14.29	5.80	14.29
		1.4	3.87	18.35	4.81	15.38
15	0.2	0.6	1.55	9.82	2.69	7.32
		1.0	1.60	10.41	1.20	8.00
		1.4	4.04	13.96	6.06	21.05
	0.5	1.0	4.26	11.19	7.72	16.67
		1.4	3.03	11.90	8.93	17.65
20	0.2	0.6	0.78	3.80	1.45	3.77
		1.0	3.16	7.38	0.30	30.30
		1.4	1.80	9.22	4.15	10.34

Secondly, we compare station paralleling algorithm with our Branch and Bound algorithm. For flexibility ratios of 0.2 and 0.5 the average percentage deviations fluctuate between 0.23 – 1.77 and 0.34 – 2.50, respectively when L/A_l is 30/1. The average percentage deviations vary between 0.45 – 2.20 and 0.67 – 2.54, respectively when L/A_l is 30/30. The maximum savings are 12.00% when L/A_l is 30/1 and 11.76% when L/A_l is 30/30. The average percentage deviations for the optimal solution without paralleling are much higher than the ones observed for the optimal solution with station paralleling. This indicates that task paralleling brings slightly larger savings in terms of total cost compared to station paralleling.

Table 5.11: Total cost of Branch and Bound task paralleling algorithm, non-paralleling option, and station paralleling algorithm.

			% Deviations											
			Non-paralleled						Station paralleling					
			L/A_l = 30/1			L/A_l = 30/30			L/A_l = 30/1			L/A_l = 30/30		
N	FR	c	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.
10	0.2	1.0	10.35	23.29	0.00	3.45	12.50	0.00	1.42	11.07	0.00	2.20	10.53	0.00
		1.4	14.62	24.10	0.00	3.40	16.67	0.00	0.23	1.05	0.00	1.18	11.76	0.00
	0.5	1.0	10.61	25.00	0.00	4.87	25.00	0.00	1.57	12.00	0.00	2.37	6.67	0.00
		1.4	14.88	24.50	0.00	4.02	17.65	0.00	0.36	1.24	0.00	0.67	6.67	0.00
15	0.2	1.0	10.13	18.71	0.00	3.27	12.50	0.00	1.08	8.14	0.00	0.45	4.55	0.00
		1.4	16.05	20.85	9.82	5.16	21.05	0.00	1.77	10.16	0.00	1.59	6.25	0.00
	0.5	1.0	8.39	15.35	0.00	3.80	14.81	0.00	0.34	0.93	0.00	1.30	8.00	0.00
		1.4	11.15	20.77	0.00	5.18	12.50	0.00	2.50	11.90	0.00	2.54	10.00	0.00
20	0.2	1.0	12.71	26.04	0.00	1.38	5.41	0.00	0.75	5.60	0.00	1.11	3.03	0.00
		1.4	13.32	20.50	8.46	4.51	15.38	0.00	1.29	9.72	0.00	1.64	9.09	0.00

Table 5.12: Total cost of non-paralleling option and heuristic algorithm.

N	FR	c	% Deviations					
			L/A_l = 30/1			L/A_l = 30/30		
			Avg.	Max.	Min.	Avg.	Max.	Min.
10	0.2	1	7.54	13.24	0.00	-1.74	5.88	-11.11
		1.4	11.23	24.10	-1.60	-1.71	16.67	-14.29
	0.5	1	5.40	22.98	-1.07	-0.95	12.50	-9.09
		1.4	11.08	23.69	-1.06	-1.11	16.67	-18.18
15	0.2	1	8.66	18.28	-0.64	2.00	12.50	-8.70
		1.4	12.17	20.14	-1.40	-1.05	5.56	-6.25
	0.5	1	4.12	14.85	-1.06	-4.56	7.41	-13.64
		1.4	8.29	20.14	-0.89	-4.17	0.00	-12.50
20	0.2	1	9.87	21.70	-1.50	1.07	5.41	0.00
		1.4	11.54	20.05	6.12	0.16	15.38	-7.41
	0.5	1	3.18	13.73	-1.50	-1.31	2.86	-6.25
		1.4	10.54	15.55	7.39	-0.40	15.38	-7.69
25	0.2	1	7.41	13.95	-0.17	-1.67	0.00	-6.98
		1.4	11.41	25.98	0.00	-0.10	2.70	-5.71
	0.5	1	4.68	9.49	-1.05	-2.15	4.76	-9.09
		1.4	9.99	21.80	-0.28	-1.15	2.94	-5.56
30	0.2	1	7.89	12.30	-0.56	-0.01	1.92	-1.96
		1.4	11.46	16.04	4.55	-1.52	5.71	-7.32
	0.5	1	5.54	14.25	-0.16	-1.30	2.13	-6.67
		1.4	8.05	15.40	-0.96	-2.05	2.33	-7.32
50	0.2	1	8.36	15.04	3.89	-0.01	2.22	-1.16
		1.4	13.25	19.82	5.81	-0.82	1.33	-5.56
	0.5	1	5.14	10.98	1.12	-1.07	0.00	-3.57
		1.4	6.16	12.22	-1.35	-2.64	0.00	-5.80

Table 5.12 reports the total cost deviation (NP – H) which is computed as (NP–H) / H × 100. Table 5.12 shows that, for flexibility ratios of 0.2 and 0.5, the average percentage deviations fluctuate between 7.41 – 13.25 and 3.18 – 11.08, respectively when L/A_l is 30/1. The maximum percentage deviation is 25.98 when L/A_l is 30/1. Hence, even with a heuristic task paralleling algorithm, we can bring in cost savings compared to non-paralleling. For the cost ratio 30/30, the heuristic solutions are greater than non-paralleled optimal solutions, which indicate that the heuristic task paralleling is not as effective for cost ratio 30/30 as it is for 30/1.

It is notable that our heuristic solution with task paralleling is much better than the optimal solution without paralleling when the unit equipment cost is much lower than the unit station opening cost. However, when the equipment cost is equal to the

station opening cost, the optimal solutions without paralleling are slightly better than the results obtained by our heuristic algorithm with exceptions indicated by large positive maximum percentage deviations for this cost ratio.

CHAPTER 6

CONCLUSIONS

We develop an exact algorithm and a heuristic algorithm for single model deterministic ALB problem with task paralleling. The objective is to minimize the number of workstations and the total equipment cost. We propose a Branch and Bound algorithm that allows two level task paralleling. We use the solutions obtained from the heuristic algorithm as upper bounds to improve the efficiency of the exact algorithm. To the best of our knowledge, our study proposes the first exact algorithm that considers station opening cost and task dependent equipment cost simultaneously by allowing task paralleling.

We conduct computational runs to analyze the effects of experimental parameters like problem size, cycle time, cost structure and flexibility ratio on the cost related and algorithm related performance measures. We compare the results of our exact algorithm with those of our heuristic algorithm results, station paralleling results and non-paralleling option in terms of cost related measures. Our exact algorithm results are also compared to station paralleling results and non-paralleling option results in terms of algorithm related measures. The performance of our heuristic solution as a percentage deviation from the non-paralleling option solution is also analyzed.

We found that the most important parameters that affect the difficulty of the solutions are the number of tasks, flexibility ratio and the cycle time.

Our Branch and Bound algorithm produces more nodes for assessing alternatives than station paralleling algorithm and non-paralleling option. Hence, the running time of our Branch and Bound algorithm is longer than the others.

Task paralleling provides about 12% savings on the average in terms of total cost over the non-paralleling option when the unit equipment cost is much lower than the unit station opening cost. Average savings is about 4% when the unit equipment cost is equal to the unit station opening cost. Task paralleling also brings about 1% cost reduction compared to station paralleling when L/A_l is 30/1 and L/A_l is 30/30. Hence, task paralleling can be thought of as an improvement over station paralleling.

The heuristic algorithm produces fairly good results. On the average deviation from the optimal solution with task paralleling is 3% when L/A_l is 30/1 and 4% when L/A_l is 30/30. In addition, heuristic algorithm performs 8% better than non-paralleling option in terms of total cost when L/A_l is 30/1. Therefore, for problem sizes greater than or equal to 20, heuristic algorithm can be used to obtain near-optimal solutions in particular.

In general, task paralleling performs slightly better than station paralleling since it allows tasks to be paralleled individually instead of paralleling all the tasks in a station. Task paralleling at least performs the same as station paralleling on the same problem.

This study can form a basis for the following future research topics:

- More than two level task paralleling may be considered which would further increase the production rate of the assembly line.
- Instead of fixed unit equipment cost, variable equipment cost for each equipment type may be taken into account. Moreover, more than one equipment type for each task can be assumed.
- The ALB problem solution may be optimized according to the given number of workstations instead of fixed cycle time (type II problem). In this case maximizing the production rate might be of interest.

REFERENCES

1. Askin, R.G., and Zhou, M., *A Parallel Station Heuristic for the Mixed Model Production Line Balancing Problem*, International Journal of Production Research, Vol. 35 No. 11 (3095-3105), 1997

2. Bard J.F., *Assembly Line Balancing with Parallel Workstations and Dead Time,* International Journal of Production Research, Vol. 27 No. 6 (1005-1018), 1989

3. Bukchin, J., and Rubinovitz, J., *A Weighted Approach for Assembly Line Design with Station Paralleling and Equipment Selection*, IIE Transactions, Vol. 35 (73-85), 2003

4. Ege, Y., *Assembly Line Balancing With Station Paralleling*, M.S. Thesis in Industrial Engineering, METU, 2001

5. Erel, E., and Sarin, S.C., *A Survey of Assembly Line Balancing Procedures*, Production Planning and Control, Vol. 9 No. 5 (414-434), 1998

6. Ghosh, S., and Gragnon, R.J., *A Comprehensive Literature Review and Analysis of the Design, Balancing and Scheduling of the Assembly Systems*, International Journal of Management Science, Vol. 27 No. 4 (637-670), 1989

7. McMullen, P.R., and Frazier, G.V., *A Heuristic for Solving Mixed Model Line Balancing Problems with Stochastic Task Durations and Parallel Stations*, International Journal of Production Economics, Vol. 51 (177-190), 1997

8. McMullen, P.R., and Frazier, G.V., *Using Simulated Annealing To Solve a Multiobjective Assembly Line Balancing Problem with Parallel Workstations*, International Journal of Production Research, Vol. 36 No. 10 (2717-2741), 1998

9. McMullen, P.R., and Tarasewich, P., *Using Ant Techniques to Solve the Assembly Line Balancing Problem*, IIE Transactions 35 (605-617), 2003

10. Nof, S.Y., Wilhelm, W.E., and Warnecke H., *Industrial Assembly*, Chapman & Hall, London, 1997

11. Pinto, P., Dannenbring, D.G., and Khumawala, B.M., *A Branch and Bound Algorithm for Assembly Line Balancing with Paralleling*, International Journal of Production Research, Vol. 13 No. 2 (183-196), 1975

12. Pinto, P., Dannenbring, D.G., and Khumawala, B.M., *Branch and Bound and Heuristic Procedures for Assembly Line Balancing with Paralleling of Stations*, International Journal of Production Research, Vol. 19 No. 5 (565-576), 1981

13. Sarker, B.R., and Shantikumar, J.G., *A Generalized Approach for Serial or Parallel Line Balancing*, International Journal of Production Research, Vol. 21 No. 1 (109-133), 1983

14. Scholl, A., *Balancing and Sequencing of Assembly Lines, 2nd Ed.*, Physica-Verlag, Heidelberg, 1999

15. Suer, G.A., *Designing Parallel Assembly Lines*, Computers & Industrial Engineering, Vol. 35 No. 3-4 (467-470), 1998

16. Vilarinho, P.M., and Simaria, A.S., *A Two-stage Heuristic Method for Balancing Mixed-model Assembly Lines with Parallel Workstations*, International Journal for Production Research, Vol. 40 No. 6 (1405-1420), 2002

APPENDIX A

ASSEMBLY LINE BALANCING WITH PARALLELING LITERATURE SUMMARY

In Tables A1 and A2, the literature survey conducted on assembly line balancing is summarized. The assumptions, objectives, approach, computational and other details are given for the references discussed in Chapter 3 of this study.

Table A.1: Literature Review on Single Model Deterministic ALB with Paralleling.

Reference	*Assumptions*	*Objective*	*Approach*	*Computations*	*Others*
A Branch and Bound Algorithm for Assembly Line Balancing with Paralleling **Pinto et al., 1975**	Deterministic task times Paced assembly line Production rate is known No equipment cost A task can be performed at more than one station	Minimizing the sum of the cost of labor (both regular and overtime costs) and the cost of replication of facilities for performing the tasks paralleled	Branch and Bound (B&B) Algorithm	A program using the min. cost fixed cost branching decision rule and offering optional use of the heuristic procedure based on the feasible cycle time tested on 10, 14 and 20 tasks problems	Task paralleling is allowed Lower and upper bounds on cycle time determine the maximum and minimum number of stations
Branch and Bound Heuristic Procedures for Assembly Line Balancing with Paralleling of Stations **Pinto et al., 1981**	Deterministic task times Paced assembly line Production rate is known No equipment cost A task can be performed at more than one station At most two parallel stations in each stage	Minimizing the sum of the cost of labor (both regular and overtime costs) and the cost of additional production facilities when paralleled stations are used (fixed cost of paralleled facilities)	B&B Algorithm and an heuristic solution procedure derived from B&B method	Computation of results for the 29 modified task problems	Two-level station paralleling is allowed

Table A.1 (continued): Literature Review on Single Model Deterministic ALB with Paralleling.

Reference	*Assumptions*	*Objective*	*Approach*	*Computations*	*Others*
A Generalized Approach for Serial or Parallel Line Balancing **Sarker and Shanthikumar, 1983**	Integer and known task times and cycle time No zoning constraint is allowed Maximum number of parallel stations and maximum slackness to be allowed at each stage are defined	Minimizing manpower idleness (phase II) To evaluate different alternatives total cost which is composed of labor cost and facility cost is used	A two-phase heuristic technique is used	Comparative results of manpower idleness in the generalized approach of paralleling based on the RPW technique and the modified M&Y technique	Station paralleling is allowed
Assembly Line Balancing with Parallel Workstations and dead Time **Bard, 1989**	Deterministic task times All stations and all units are identical There is an equipment cost for each task System cost for each station is the same for all stations	Minimizing the sum of the cost of equipment for each task, system cost for each station and cost of operating facilities	Dynamic Programming	15 problems are examined with number of tasks ranging from 15 to 40	Two-level station paralleling is allowed Unproductive or dead time at a station is taken into account

Table A.1 (continued): Literature Review on Single Model Deterministic ALB with Paralleling.

Reference	*Assumptions*	*Objective*	*Approach*	*Computations*	*Others*
Designing Parallel Assembly Lines **Suer ,1998**	Deterministic task times	Minimize total manpower	A Heuristic Algorithm based on a 3-phase methodology		Tasks are grouped into stations for various configurations, and then the number of parallel stations is determined by a model that maximizes assembly rate.
Assembly Line Balancing with Station Paralleling **Ege, 2001**	An arbitrary number of parallel stations can be assigned to each stage Cycle time is given There is a fixed cost for opening a station Precedence relations are known and fixed	Minimize the total cost that includes station opening and equipment cost	An optimal seeking Branch and Bound Algorithm, A Heuristic Branch and Bound Algorithm	The algorithm is tested with 3 level station paralleling on problems having tasks ranging from 10 to 70 with cycle times of 1, 1.4 and 1.8 times of the maximum task time	Station paralleling The effect of equipment cost is tested with equipment costs of 1/30 and 30/30 of the station opening cost

Table A.1 (continued): Literature Review on Single Model Deterministic ALB with Paralleling.

Reference	***Assumptions***	***Objective***	***Approach***	***Computations***	***Others***
A Weighted Approach for Assembly Line Design with Station Paralleling and Equipment Selection **Bukchin and Rubinovitz, 2003**	Deterministic task times, cycle time is given, station paralleling is allowed	Minimize the number of stations incorporating cost/weight factor for different paralleling situations	Branch and Bound algorithm	The effects of the flexibility ratio and maximal number of parallel stations on the balancing improvement are tested	Station paralleling

Table A.2: Literature Review on Mixed Model ALB with Paralleling.

Reference	*Assumptions*	*Objective*	*Approach*	*Computations*	*Others*
A Parallel Station Heuristic for the Mixed- Model Production Line Balancing Problem **Askin and Zhou, 1997**	Constant and known demand rate, operation processing times and precedence constraints for each product Task times are deterministic Arbitrary number of identical, parallel workstation at each stage	Minimizing the sum of the fixed cost for operating stations and total tooling cost	An Heuristic Algorithm	8 problems with tasks 30 or 100, also the 25 task problem in Bard (1989) are solved with the heuristic algorithm (comparison on the computation times and iteration number)	Parallel stations Task dependent equipment costs Station utilization is considered by using a threshold variable for acceptable levels
A Heuristic for Solving Mixed-Model Line Balancing Problems with Stochastic Task Durations and Parallel Stations **McMullen and Frazier, 1997**	Stochastic task times Cycle time is given Number of parallel stations is unrestricted Changeover times between products are negligible	Output performance measures like average WIP inventory level, units of throughputs,% of units been completed within cycle time, average system utilization,% of desired cycle time with 10 different task selection rules	Modified incremental utilization heuristic	6 different problems with 21, 25, 29, 40, 45, and 74 tasks and 10 task rules; totally 60 layouts were simulated (25 simulation runs)	Station paralleling (by allowing multiple workers within cells) Some performance measures are highly correlated

Table A.2 (continued): Literature Review on Mixed Model ALB with Paralleling.

Reference	*Assumptions*	*Objective*	*Approach*	*Computations*	*Others*
Using Simulated Annealing to Solve a Multiobjective Assembly Line Balancing Problem with Parallel Workstations **McMullen and Frazier, 1998**	Stochastic task times Parallel stations are allowed Cycle time is specified All workers on the assembly line posses the same level of skill Changeover times between products are negligible	Minimize labor and equipment cost Minimize smoothness index Minimize probability of lateness Minimize composite function (combinations of above objectives with different weight factors)	Simulated Annealing	7 different line balancing problems (11, 21, 25, 29, 40, 45 and 74 tasks) were each solved by 23 different heuristics (6 of them use simulated annealing) yielding 161 different assembly line layouts for analysis	Parallel workstations Multiple objective
A two-stage Heuristic Method for Balancing Mixed-Model Assembly Line with Parallel Workstations **Vilarinho and Simaria, 2002**	Cycle time is known Paced assembly line A set of similar models can be simultaneously assembled	Minimizing the number of workstations before the secondary goal becomes active (balancing the workload between and within workstations)	A two-stage simulated annealing	Tested on a set of 20 problems with number of tasks 8 to 70	Parallel workstations An upper bound on the maximum number of replicas of a workstation can be set by decision maker

Table A.2 (continued): Literature Review on Mixed Model ALB with Paralleling.

Reference	*Assumptions*	*Objective*	*Approach*	*Computations*	*Others*
Using Ant Techniques To Solve the Assembly Line Balancing Problem **McMullen and Tarasewich, 2003**	Task times are stochastic Cycle time is specified Station paralleling is allowed	According to the objective function there are four versions of the algorithm: maximizing utilization of assembly line layout, maximizing the probability of all work centers being completed on time, maximizing the composite measure of utilization and probability of on time completion, minimizing design cost	A heuristic algorithm that uses Ant Colony Optimization (ACO) techniques	On problems having 21 – 74 tasks the output performance measures resulting from this approach are compared to the ones obtained from 23 other heuristic methods	Station paralleling

APPENDIX B

EXAMPLE PROBLEM SOLUTION FOR BRANCH AND BOUND ALGORITHM

Initialization phase of the solution procedure with Branch and Bound algorithm for example problem III is given in Section 4.3.1. The detailed calculations are given below.

Remove node 1 from stack

$S = \varnothing$

$ST_1 = \{1\}$, $SE_1 = \{1\}$, $SP_1 = \varnothing$, $RT_1 = 7 - 3 = 4$, $UT = \{2,3,4,5,6\}$

$TC_1 = 0$

Min. required number of stations = (8 + 2 + 3 + 4 + 5) / 7 = 3.14 (rounded up to 4)

For calculating minimum remaining equipment cost, the required number of each equipment type is found as follows.

For equipment type 1; (7 / 7) = 1

For equipment type 2; (2 / 7) = 0.29 (rounded up to 1)

For equipment type 3; (8 / 7) = 1.14 (rounded up to 2)

For equipment type 4; (3 / 7) = 0.43 (rounded up to 1)

For equipment type 5; (5 / 7) = 0.71 (rounded up to 1)

Total number of equipment pieces needed = 1 + 1 + 2 + 1 + 1 = 6

Min. remaining equipment cost = $6 \times 5 = 30$

$LB_1 = 0 + (4 \times 50) + 30 = 230$

Is $UT = \varnothing$? NO

Possible subnodes are created for node 1 considering that tasks 2 and 3 can be assigned next.

Node 3: Task 2 is assigned to station 1 without paralleling.

$TC_3 = 0$

$LB_3 = 0 + (4 \times 50) + (6 \times 5) = 230$

Create this node as $LB_3 < UB$.

Node 4: Task 2 is assigned to station 1 with paralleling.

$TC_4 = 0$

$LB_4 = 0 + (4 \times 50) + (7 \times 5) = 235$

Create this node as $LB_4 < UB$.

Node 5: Task 2 is assigned to station 2 without paralleling.

$TC_5 = (1 \times 50) + (1 \times 5) = 55$

$LB_5 = 55 + (4 \times 50) + (6 \times 5) = 285$

Fathom this branch, as $LB_5 \geqslant UB$

Node 6: Task 2 is assigned to station 2 with paralleling.

$TC_6 = (1 \times 50) + (1 \times 5) = 55$

$LB_6 = 55 + (4 \times 50) + (7 \times 5) = 290$

Fathom this branch, as $LB_6 \geqslant UB$

Node 7: Task 3 is assigned to station 1 with paralleling.

(Note that time of task 3 is longer than the cycle time and so task 3 has to be paralleled.)

$TC_7 = 0$

$LB_7 = 0 + (4 \times 50) + (6 \times 5) = 230$

Create this node as $LB_7 < UB$

Node 8: task 3 is assigned to station 2 with paralleling.

$TC_8 = (1 \times 50) + (1 \times 5) = 55$

$LB_8 = 55 + (4 \times 50) + (6 \times 5) = 285$

Fathom this branch, as $LB_8 \geqslant UB$

Is the number of subnodes > 0? YES

Subnodes 7, 3, 4 are added to the stack in given order according to their lower bound values. Node 7 stays at the top of the stack for next cycle since it has the lowest lower bound value. Figure B.1 illustrates the situation after subnodes of node 1 are created.

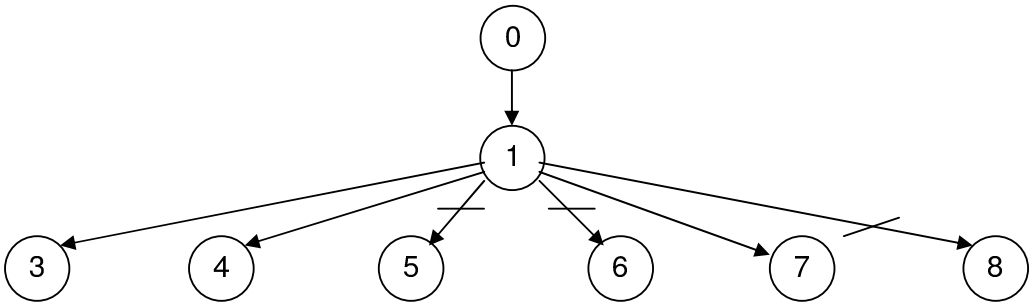

Figure B.1: Branching after node 1 for example problem III.

Remove node 7 from stack

$S = \{3, 4\}$

$ST_1 = \{1, 3\}$, $SE_1 = \{1, 3\}$, $SP_1 = \{3\}$, $RT_1 = 4 - 4 = 0$, $UT = \{2,3,4,5,6\}$

Is $UT = \varnothing$? NO

Possible subnodes are created for node 7. Since station 1 is closed with zero remaining time, only assignments to station 2 are considered.

Node 9: Task 2 is assigned to station 2 without paralleling.

$TC_9 = (1 \times 50) + (2 \times 5) = 60$

$LB_9 = 60 + (3 \times 50) + (5 \times 5) = 235$

Create this node as $LB_9 < UB$

Node 10: Task 2 is assigned to station 2 with paralleling.

$TC_{10} = (1 \times 50) + (2 \times 5) = 60$

$LB_{10} = 60 + (3 \times 50) + (6 \times 5) = 240$

Fathom this branch, as $LB_{10} \geqslant UB$

Node 11: Task 3 is assigned to station 2 with paralleling.

$TC_{11} = (1 \times 50) + (2 \times 5) = 60$

$LB_{11} = 60 + (3 \times 50) + (5 \times 5) = 235$

Create this node as $LB_{11} < UB$

Is the number of subnodes > 0? YES

Subnodes 9, 11 are added to the stack in given order. Figure B.2 illustrates the situation after subnodes of node 7 are created.

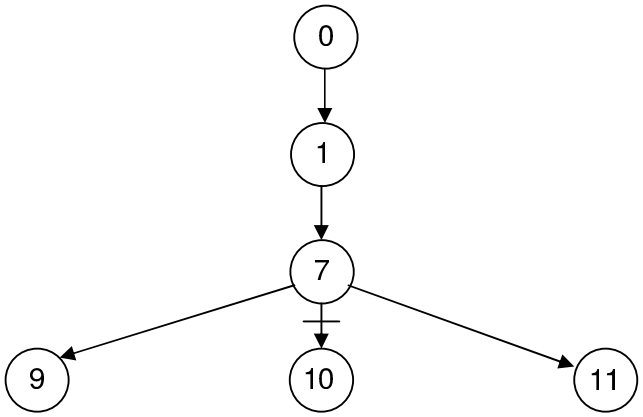

Figure B.2: Branching after node 7 for example problem III.

Remove node 9 from stack

S = {3,4,11}

$ST_2 = \{2\}$, $SE_2 = \{2\}$, $SP_2 = \varnothing$, $RT_2 = 7 - 2 = 5$, UT = {3,4,5,6}

Is UT = ∅? NO

Possible subnodes are created for node 9. Because of the precedence relations task 3 and 5 are considered to be assigned to station 2 and 3.

Node 12: Task 3 is assigned to station 2 with paralleling.

$TC_{12} = (1 \times 50) + (2 \times 5) = 60$

$LB_{12} = 60 + (3 \times 50) + (5 \times 5) = 235$

Create this node as $LB_{12} < UB$

Node 13: Task 3 is assigned to station 3 with paralleling.

$TC_{13} = (2 \times 50) + (3 \times 5) = 115$

$LB_{13} = 115 + (3 \times 50) + (4 \times 5) = 285$

Fathom this branch, as $LB_{13} \geqslant UB$

Node 14: Task 5 is assigned to station 2 without paralleling.

$TC_{14} = (1 \times 50) + (2 \times 5) = 60$

$LB_{14} = 60 + (3 \times 50) + (5 \times 5) = 235$

Create this node as $LB_{14} < UB$

Node 15: Task 5 is assigned to station 2 with paralleling.

$TC_{15} = (1 \times 50) + (2 \times 5) = 60$

$LB_{15} = 60 + (3 \times 50) + (6 \times 5) = 240$

Fathom this branch, as $LB_{15} \geqslant UB$

Node 16: Task 5 is assigned to station 3 without paralleling.

$TC_{16} = (2 \times 50) + (3 \times 5) = 115$

$LB_{16} = 115 + (3 \times 50) + (4 \times 5) = 285$

Fathom this branch, as $LB_{16} \geqslant UB$

Node 17: Task 5 is assigned to station 3 with paralleling.

$TC_{17} = (2 \times 50) + (3 \times 5) = 115$

$LB_{17} = 115 + (3 \times 50) + (5 \times 5) = 290$

Fathom this branch, as $LB_{17} \geqslant UB$

Is the number of subnodes > 0? YES

Subnodes 12, 14 are added to the stack in given order. Figure B.3 illustrates the situation after subnodes of node 9 are created.

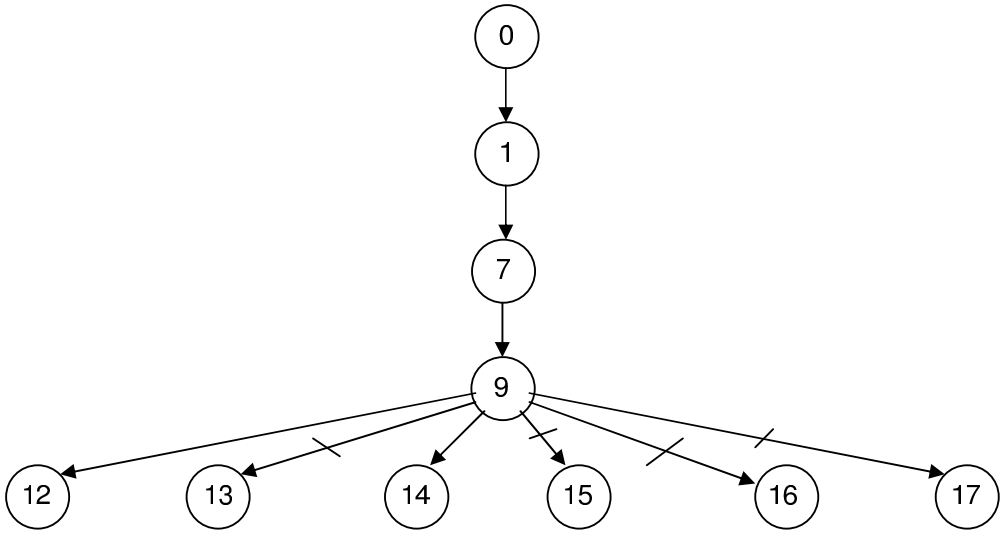

Figure B.3: Branching after node 9 for example problem III.

The algorithm goes on according to the flowchart in Figure 4.4 and finds the optimal solution as 235.

APPENDIX C

COMPUTATIONAL RESULT TABLES

Table C.1: Cost related measures for station paralleling algorithm when L/A_l is 30/1.

N	FR	c	Total Cost, TC		# of Stations, NS	
			Avg.	Max.	Avg.	Max.
10	0.2	1.0	222900	283000	7.1	9
		1.4	156000	193000	4.9	6
	0.5	1.0	213300	280000	6.8	9
		1.4	155500	193000	4.9	6
15	0.2	1.0	326800	382000	10.4	12
		1.4	230900	261000	7.2	8
	0.5	1.0	307000	378000	9.7	12
		1.4	229300	285000	7.2	9
20	0.2	1.0	432500	509000	13.6	16
		1.4	311100	353000	9.7	11
	0.5	1.0	407900	482000	12.8	15
		1.4	296600	350000	9.2	11
25	0.2	1.0	529200	593000	16.6	19
		1.4	377900	426000	11.6	13
	0.5	1.0	505800	593000	15.8	19
		1.4	367000	423000	11.2	13
30	0.2	1.0	636100	697000	20.0	22
		1.4	449800	494000	13.8	15
	0.5	1.0	608800	692000	19.1	22
		1.4	439600	486000	13.5	15
50	0.2	1.0	1043900	1138000	32.5	36
		1.4	746400	816000	22.5	25
	0.5	1.0	1030500	1137000	32.3	36
		1.4	721200	800000	21.9	25

Table C.2: Cost related measures for station paralleling algorithm when L/A_l is 30/30.

N	FR	c	Total Cost, TC		# of Stations, NS	
			Avg.	Max.	Avg.	Max.
10	0.2	1.0	492000	600000	7.8	10
		1.4	387000	510000	5.8	8
	0.5	1.0	468000	570000	7.4	9
		1.4	372000	480000	5.6	7
15	0.2	1.0	732000	900000	11.1	15
		1.4	609000	720000	8.3	10
	0.5	1.0	702000	840000	10.6	13
		1.4	594000	720000	7.8	9
20	0.2	1.0	1029000	1170000	15.7	19
		1.4	831000	990000	10.6	13
	0.5	1.0	963000	1140000	14.2	18
		1.4	801000	900000	10.5	12
25	0.2	1.0	1275000	1410000	18.8	22
		1.4	1107000	1200000	13.8	17
	0.5	1.0	1233000	1320000	17.8	20
		1.4	1062000	1170000	13.4	16
30	0.2	1.0	1527000	1620000	22.7	25
		1.4	1299000	1470000	16.7	19
	0.5	1.0	1479000	1620000	21.8	25
		1.4	1239000	1380000	15.7	18
50	0.2	1.0	2598000	2640000	37.5	40
		1.4	2274000	2310000	27.8	32
	0.5	1.0	2559000	2670000	35.9	40
		1.4	2211000	2250000	26	29

Table C.3: Algorithm related measures for station paralleling algorithm when L/A_l is 30/1.

N	FR	c	CPU Time		# of Nodes		Optimum Node #	
			Avg.	Max.	Avg.	Max.	Avg.	Max.
10	0.2	1.0	0	0	127.8	268	27.4	83
		1.4	0	0	74.7	148	18.3	60
	0.5	1.0	0	0	1529.8	6274	188.8	1231
		1.4	0.0002	0.0008	525.7	1668	77.6	357
15	0.2	1.0	0.0002	0.0007	2416.6	9849	1490.1	8717
		1.4	0.5	2	1124.5	3615	582.1	2077
	0.5	1.0	0.0004	0.0012	5594.5	14107	1779.5	5187
		1.4	0.0011	0.0022	4954.4	12582	876.1	5242
20	0.2	1.0	0.0023	0.0108	25035	109917	4475.3	31883
		1.4	0.0007	0.0017	4125	11637	681	3313
	0.5	1.0	0.0171	0.1275	185830.6	1342771	2734	9775
		1.4	0.0035	0.02	27591.1	158965	3175.3	20393
25	0.2	1.0	0.0096	0.0495	78938.4	411897	27495.2	222684
		1.4	0.002	0.0075	9284.9	29356	4056.1	15172
	0.5	1.0	0.0959	0.7903	1072081	8942624	849957.8	7777600
		1.4	0.01511	0.0369	110441.9	353227	48952	148323
30	0.2	1.0	0.0881	0.2582	629021.8	1801597	314119.5	1715627
		1.4	0.0061	0.0145	46933.8	192186	24127.6	190485
	0.5	1.0	6.7869	54.9928	60876842	497235262	29861333	283174709
		1.4	0.20969	0.5378	1482695	4123987	502209.6	1971079
50	0.2	1.0	35.31902	117.9725	191748402	620968768	63749371	239568560
		1.4	1.35644	5.0986	3764499	15890523	2056519.7	7992350
	0.5	1.0	180.0305	180.0712	957391501	1326000000	475326197	1109362450
		1.4	75.05989	180.0143	220040794	816500000	100523562	233608871

Table C.4: Algorithm related measures for station paralleling algorithm when L/A_l is 30/30.

N	FR	c	CPU Time		# of Nodes		Optimum Node #	
			Avg.	Max.	Avg.	Max.	Avg.	Max.
10	0.2	1.0	0.0000	0.0002	164	415	13.7	46
		1.4	0.1000	1	107.8	206	20.2	73
	0.5	1.0	0.0001	0.0005	1562.8	8597	175.8	736
		1.4	0.0003	0.0008	702.4	2378	142.6	817
15	0.2	1.0	0.0003	0.0017	3454.2	20041	1962.3	18234
		1.4	0.0013	0.0108	2064.4	8630	549.9	2433
	0.5	1.0	0.0009	0.0033	13247.6	43935	3833.9	13248
		1.4	0.0015	0.0058	9151.5	36348	2882.8	25860
20	0.2	1.0	0.0050	0.0192	54398	229593	15991.4	159396
		1.4	0.0024	0.0097	22781.5	136750	1858.5	5515
	0.5	1.0	0.0598	0.435	714819.1	5265827	1473.8	7490
		1.4	0.0159	0.0533	185105.9	648355	9281.5	58403
25	0.2	1.0	0.0057	0.0308	55411.1	282748	3250	30935
		1.4	0.0061	0.0233	68204.9	210421	9010.5	50560
	0.5	1.0	0.4699	3.9048	6187955	52537967	4643658.4	44422638
		1.4	0.0702	0.3338	715791.1	3148995	287251.3	1873206
30	0.2	1.0	0.2422	0.6165	1843920	4565624	135590.6	1242321
		1.4	0.4274	1.8631	1845738	7519709	477599.1	1751561
	0.5	1.0	21.9512	93.8548	192835024	741974801	8300252.5	32636997
		1.4	54.1522	180.0248	127258974	418500001	5229852.3	20546225
50	0.2	1.0	87.5255	180.0882	492932143	1088000000	8598231.4	79928084
		1.4	123.8396	180.1672	478062578	895000001	16435467	94061131
	0.5	1.0	180.0213	180.0608	1443100002	1712000000	4366533.7	43649478
		1.4	180.0133	180.0857	886850001	1516500001	63192971	383227389

Table C.5: Cost related measures for non-paralleling option when L/A_l is 30/1.

N	FR	c	Total Cost, TC		# of Stations, NS	
			Avg.	Max.	Avg.	Max.
10	0.2	1.0	245700	310000	7.9	10
		1.4	184300	250000	5.9	8
	0.5	1.0	236400	280000	7.6	9
		1.4	184000	249000	5.9	8
15	0.2	1.0	361500	465000	11.6	15
		1.4	270400	313000	8.6	10
	0.5	1.0	334300	405000	10.7	13
		1.4	252300	285000	8.0	9
20	0.2	1.0	492600	590000	15.8	19
		1.4	356200	439000	11.3	14
	0.5	1.0	447200	560000	14.3	18
		1.4	343300	403000	10.9	13
25	0.2	1.0	587700	685000	18.8	22
		1.4	437300	562000	13.8	18
	0.5	1.0	563700	624000	18.0	20
		1.4	424200	532000	13.4	17
30	0.2	1.0	715300	777000	22.9	25
		1.4	524900	600000	16.6	19
	0.5	1.0	684600	779000	21.9	25
		1.4	497500	569000	15.7	18
50	0.2	1.0	1177200	1279000	37.6	41
		1.4	881800	1004000	27.8	32
	0.5	1.0	1126100	1248000	35.9	40
		1.4	806400	884000	25.3	28

Table C.6: Cost related measures for non-paralleling option when L/A_l is 30/30.

N	FR	c	Total Cost, TC		# of Stations, NS	
			Avg.	Max.	Avg.	Max.
10	0.2	1.0	498000	600000	7.9	10
		1.4	396000	540000	5.9	8
	0.5	1.0	480000	570000	7.7	9
		1.4	387000	510000	5.9	8
15	0.2	1.0	753000	900000	11.6	15
		1.4	630000	720000	8.6	10
	0.5	1.0	720000	840000	10.7	13
		1.4	609000	720000	8.1	9
20	0.2	1.0	1032000	1170000	15.8	19
		1.4	855000	990000	11.3	14
	0.5	1.0	975000	1140000	14.3	18
		1.4	816000	930000	10.9	13
25	0.2	1.0	1275000	1410000	18.8	22
		1.4	1113000	1200000	13.9	18
	0.5	1.0	1251000	1320000	18	20
		1.4	1068000	1170000	13.6	17
30	0.2	1.0	1536000	1620000	22.9	25
		1.4	1305000	1470000	16.7	19
	0.5	1.0	1485000	1620000	21.9	25
		1.4	1260000	1410000	16.1	18
50	0.2	1.0	2604000	2700000	37.6	41
		1.4	2268000	2310000	27.8	32
	0.5	1.0	2553000	2640000	36	40
		1.4	2175000	2250000	26	30

Table C.7: Algorithm related measures for non-paralleling option when L/A_l is 30/1.

N	FR	c	CPU Time		# of Nodes		Optimum Node #	
			Avg.	Max.	Avg.	Max.	Avg.	Max.
10	0.2	1.0	0.0000	0.0000	20.7	39	2.9	29
		1.4	0.0000	0.0000	36.6	103	9.7	36
	0.5	1.0	0.0001	0.0008	135.8	458	30.8	142
		1.4	0.0000	0.0000	200	836	29.5	131
15	0.2	1.0	0.2000	1.0000	161.7	608	26.3	85
		1.4	2.2000	5.0000	404	1087	79.6	333
	0.5	1.0	0.0002	0.0008	659.7	1017	147.3	513
		1.4	0.0003	0.0011	1837.1	8733	273.6	1508
20	0.2	1.0	0.0008	0.0055	1192.7	9162	476.5	4736
		1.4	0.0006	0.0025	1460.5	498	202.7	753
	0.5	1.0	0.0019	0.0078	7428.4	40914	43.9	344
		1.4	0.0016	0.0036	6038.1	19492	950.4	4617
25	0.2	1.0	0.0005	0.0027	2031.8	7872	19.7	103
		1.4	0.0011	0.0039	4400.9	13164	728.7	5603
	0.5	1.0	0.0308	0.2315	181645.4	1326116	101135.8	984052
		1.4	0.0165	0.0917	109047.1	566043	48884.8	284717
30	0.2	1.0	0.0024	0.0114	6219.3	28287	138.3	576
		1.4	0.0089	0.0200	18662.1	38886	3611.2	12979
	0.5	1.0	0.1075	0.3020	749515.1	3573665	203553.1	1767655
		1.4	0.2822	1.6752	872853.6	4988313	119557.2	937668
50	0.2	1.0	0.1844	0.3758	361383.7	809255	2273.2	22668
		1.4	3.1451	11.5481	8734423	3545953	223602.8	1751567
	0.5	1.0	124.0429	180.0248	309333751	1003500000	70067963.0	650112917
		1.4	64.3547	180.0243	252512555	754000000	14518430.0	92522153

Table C.8: Algorithm related measures for non-paralleling option when L/A_l is 30/30.

N	FR	c	CPU Time		# of Nodes		Optimum Node #	
			Avg.	Max.	Avg.	Max.	Avg.	Max.
10	0.2	1.0	0.0000	0.0000	21.6	39	2.9	29
		1.4	0.0000	0.0000	37.8	107	9.7	36
	0.5	1.0	0.0002	0.0008	193.6	619	38.1	215
		1.4	0.0000	0.0000	201.1	866	28.9	125
15	0.2	1.0	0.0003	0.0014	178.9	608	30.4	107
		1.4	0.0012	0.0017	538.4	1115	81.3	311
	0.5	1.0	0.0003	0.0011	1136.9	2560	193.5	507
		1.4	0.0006	0.0019	2169.4	7771	408.8	2924
20	0.2	1.0	0.0005	0.0031	1282.2	8994	471.7	4688
		1.4	0.0007	0.0028	2047.8	10251	389.5	2189
	0.5	1.0	0.0022	0.0089	9964.9	48820	80.2	478
		1.4	0.0018	0.0028	8091.2	15920	944.8	4204
25	0.2	1.0	0.0012	0.0090	3152.2	17797	19.7	103
		1.4	0.0027	0.0094	12262.9	44040	1462.8	10735
	0.5	1.0	0.0465	0.2800	282140.9	1656740	112092.2	1084288
		1.4	0.0370	0.2200	252920.3	1385397	111777.3	777998
30	0.2	1.0	0.0031	0.0072	9078.7	37723	201.4	972
		1.4	0.6914	6.5000	109549.2	218802	15583.3	96551
	0.5	1.0	0.8335	2.3773	1571763	3219577	174521.5	812190
		1.4	3.5709	27.8823	7311064	41955276	379022.8	1185248
50	0.2	1.0	0.2992	0.6569	749362	1642351	7101.2	70948
		1.4	24.7580	82.2419	75592610	258513617	1987314.4	11525787
	0.5	1.0	159.7280	180.0533	324321847	478554182	17892217.0	178921975
		1.4	180.0098	180.0297	656600001	1191500000	117864064.0	444622810