ASSEMBLY LINE BALANCING WITH STATION PARALLELING

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

BY

YUNUS EGE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INDUSTRIAL ENGINEERING

OCTOBER 2001

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Tayfur Öztürk
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.

Assoc. Prof. Dr. Nur Evin Özdemirel
Co-Supervisor

Assoc. Prof. Dr. Meral Azizoğlu
Supervisor

Examining Committee Members

Prof. Dr. Ömer Kırca

Assoc. Prof. Dr. Canan Sepil

Assoc. Prof. Dr. Meral Azizoğlu

Assoc. Prof. Dr. Nur Evin Özdemirel

Assoc. Prof. Dr. Cemal Akyel

ABSTRACT

ASSEMBLY LINE BALANCING WITH STATION PARALLELING

EGE, Yunus

M.Sc., Department of Industrial Engineering
Supervisor: Assoc. Prof. Dr. Meral AZİZOĞLU
Co-Supervisor: Assoc. Prof. Dr. Nur Evin ÖZDEMİREL

October 2001, 92 pages

In this study, single model assembly line balancing problem is considered. The objective is the minimization of total station opening and equipment cost. A branch and bound algorithm that gives the optimal solution is proposed to solve the problem. The most important feature of the proposed method is that task dependent equipment cost and station paralleling are considered simultaneously. Any level of station paralleling is allowed. A heuristic method available in the literature is used to determine an initial upper bound for the problem. Computational analysis is conducted to investigate the effects of changing certain parameter values on some performance measures. The proposed algorithm is also compared with the heuristic method in terms of total cost.

Keywords: Single Model Assembly Line Balancing, Station Paralleling.

ÖZ

İSTASYON PARALELLEŞTİRME İLE MONTAJ HATTI DENGELEME

EGE, Yunus

Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Meral AZİZOĞLU
Yardımcı Tez Yöneticisi: Doç. Dr. Nur Evin ÖZDEMİREL

Ekim 2001, 92 sayfa

Bu çalışmada tek modelli montaj hattı dengeleme problemi dikkate alınmıştır. Amaç toplam istasyon açma ve ekipman maliyetinin enazlanmasıdır. Problemi çözmek için optimal sonuç veren bir dal-budak algoritması önerilmektedir. Önerilen metodun en önemli özelliği görev bağımlı ekipman maliyetinin ve istasyon paralelleştirmenin eş zamanlı olarak göz önüne alınmasıdır. Herhangi bir seviyede istasyon paralelleştirmeye izin verilmektedir. Problem için bir başlangıç üst limiti belirlenmesinde literatürde bulunan bir sezgisel metod kullanılmaktadır. Belirli parametre değerlerini değiştirmenin bazı performans ölçütleri üzerindeki etkilerinin incelenmesi için sayısal analiz yapılmıştır. Önerilen algoritma ayrıca toplam maliyet açısından sezgisel metod ile karşılaştırılmıştır.

Anahtar kelimeler: Tek Modelli Montaj Hattı Dengeleme, İstasyon Paralelleştirme.

To my family

ACKNOWLEDGEMENTS

I would like to express my gratitude to Assoc. Prof. Dr. Meral Azizoğlu and Assoc. Prof. Dr. Nur Evin Özdemirel for their valuable and patient supervision and continual support through the course of this study. This study could not have been completed without their support and guidance.

I would like to express my deepest thanks to my family for their continued patience and understanding. The completion of this task would not have been possible without their endless love and faith in me. I also thank to my brother very much because of his assistance in typing my thesis.

Special thanks to my friend Burak Nalçacı because of his assistance during my thesis. I also acknowledge the continuous motivation by Banu Soylu, Ercihan Toprakçı and other my friends who listened my complaints and encouraged me to finish this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

CHAPTER 1

INTRODUCTION

Assembly line balancing problem has been studied extensively since the 1950s. The basic assembly line problem consists of assigning the tasks to an ordered sequence of workstations, such that the some prespecified restrictions are satisfied and some measure(s) of effectiveness is(are) optimized.

The traditional symbol of automation is the mechanized flow line. An automated flow line consists of several machines or workstations, which are linked together by an automated material handling mechanism. The transfer of workparts occurs automatically. Manual stations might also be located along the flow line to perform certain operations that are inefficient or infeasible to automate.

Automated flow lines are generally the most appropriate means of production in cases of relatively stable product life; high product demand, which requires high production rates. The main objectives of the use of flow line automation are to reduce labor cost, to increase production rates, to reduce work in process inventory.

In an automated flow line, achieving an even distribution of work among workstations is highly desirable. This can be achieved by line balancing. The topic of line balancing has been of great interest to academicians. There exists a rich literature on this topic. But the studies on line balancing with paralleling are relatively new. Paralleling brings flexibility to the design of flow lines. By

paralleling, total facility cost may further be reduced and the production volume can be increased.

There are two types of paralleling: paralleling of tasks and paralleling of stations. Paralleling a task allows that task to be performed at more than one station. Since each paralleled task must be performed at more than one station, additional production facilities would be required.

In paralleling a station, all production facilities at that station are duplicated. Station paralleling can reduce idle time by fitting more tasks to stations because the workcontent allowed at each paralleled station is equal to the cycle time multiplied by the number of times a station is paralleled.

Most of the solution procedures in the literature assume that the longest task time can not be greater than the predetermined cycle time. Relaxing of this assumption by allowing paralleling of tasks or paralleling of stations enables to solve the problems with task times greater than the cycle time.

Although substantial benefits of paralleling, when paralleling is a possibility, the trade-off between the additional investment required for parallel stations and the labor savings derived from paralleling must be considered. However, in many cases paralleling is a less costly alternative for increasing the production compared to other alternatives like overtime, subcontracting and buffer stocks.

The purpose of this study is to develop an exact algorithm for single model assembly line balancing which minimizes the total cost. Total cost has two components: task dependent equipment cost and station opening costs. Our approach includes station paralleling at any level and considers task dependent equipment cost and station paralleling cost simultaneously.

Most of the studies involving paralleling consider single model assembly lines. The exact algorithms are rather few and consider only maximum 2-level

paralleling. "n-level paralleling" term indicates that a station is paralleled such that at most n number of parallel stations exist. According to the best of our knowledge, there is no exact algorithm that considers task dependent equipment cost and station paralleling simultaneously at any level of paralleling. Therefore, the main motivation of our study for working on assembly line balancing with paralleling stems from the hope of filling a gap in literature.

The thesis includes six chapters. In Chapter 2, the terminology used in assembly line balancing is given. Then the assembly line balancing problem is defined and the paralleling concept is introduced. In Chapter 3, we report the literature on assembly line balancing problem emphasizing the studies that are most relevant to ours. After the mathematical formulation is presented and the details of our algorithm are discussed in Chapter 4, the computational analysis and the discussion of the results are presented in Chapter 5. We conclude in Chapter 6 by our main results and suggestions for further research directions.

CHAPTER 2

AN OVERVIEW OF ASSEMBLY LINES

When a product or a family of technologically similar products exhibit high volume and stable demand over lengthy periods of time, it becomes economical to dedicate a special facility to the product or family of products under consideration. To reduce work-in-process inventory caused by such non-productive times as loading, unloading and transportation between successive operations, the workstations are physically arranged in a contigonous sequence without violating the technological ordering of manufacturing stages. The resulting facility is called a *fabrication line*, if the production process is fabrication; or *assembly line*, if it is assembly (Hax and Candea, 1984), or flow line in general.

Assembly lines rely heavily on the *Principle of Interchangeability* and the *Division of Labor*. Developed in the 1800's, the principle of interchangeability states that the individual components that make up a finished product should be interchangeable between product units. Division of labor embraces the concepts of work simplification, standardization, and specialization. Following these principles, complex activities are subdivided into elemental tasks, detailed work instructions are produced for rationally accomplishing each of these tasks independently, and the elemental tasks are assigned to different workers, who quickly become proficient in performing their repetitive operations. Together, interchangeability and division of labor facilitate mass production, allow replacement parts to be used to lengthen a product's useful life, and make the pioneering work of Henry Ford and others possible in developing assembly lines (Askin and Standridge, 1993).

There are several ways of classifying assembly lines:

- Manual versus Automated (Nature of work)
- Mechanical versus Non-mechanical (Transfer types)
- Single Model versus Multi-mixed Model (Model types)
- Straight versus U-lines (Layout)
- Parallel versus Single
- Paced versus Unpaced

According to the nature of the work, there are basically two kinds of assembly lines: *manual assembly lines* and *automated assembly lines*. *Manual assembly lines* consist of multiple workstations until the product reaches the final workstation and leaves the line as a finished good. At each workstation a portion of the total work content is performed by one or more operators. In the *automated assembly lines* the work at the stations is performed automatically rather than using operators and/or transfers between stations are automatic.

There are two methods of work transfer between workstations: by using *non-mechanical lines*, and by using *moving conveyor lines*. In non-mechanical lines, the parts are passed from one station to another by hand. In moving conveyor lines, the parts are moved between workstations by using a moving conveyor.

There are three kinds of assembly lines according to the model variations:

1. Single-model line: A single model or a product is produced in this kind of lines. The critical point of this type assembly line is that the demand rate for the product is sufficiently large.
2. Multi-model line: Two or more models, which have similar sequence of processing or assembly operations, are produced in batches on the line in this type of assembly line. While switching from one type to another, the line is rearranged.
3. Mixed-model line: In this type of line, various models are intermixed on the line so that several different models are being produced simultaneously without rearranging the line.

Traditional assembly lines are arranged in a "straight" line. Recently many new production lines are being arranged in "U-lines" rather than straight lines, as a consequence of the use of just-in-time production principles. JIT layout objectives should include reduced pick-up time, training time, overall manufacturing floor space, material handling effort and production lead time, and an increase in the production-scheduling flexibility. According to Monden (1983), U-shaped layouts are preferable for JIT systems. The most remarkable advantage of this layout is the flexibility to increase or decrease the necessary number of workers when adapting to the changes in demand. Some advantages of U-lines over traditional straight lines are the improved communication because of the close proximity of operators to each other, multiskilled operators, easier rebalancity of the change of balancing the line with fewer stations according to a traditional line (Miltenburg and Wijngaard, 1994).

If the processing time of some tasks exceeds the cycle time, Parallel or Multiple Stations can be used to balance the line. By paralleling, a more balanced line with fewer stations can be achieved.

A choice between a paced and unpaced line must be made. In a paced (synchronous) line, each workstation is given exactly the same amount of time to operate on each unit of product. At the end of this time (cycle time), the handling system automatically indexes each unit to the next station. In an unpaced (asynchronous) line, the station removes a new unit from the handling system as soon as it has completed the previous unit, performs the required tasks, and then forwards the unit on to the next station. Unlike paced lines, parts need not be passed on incomplete.

Assembly lines have the ability to keep direct labor (or automated machines) busy doing productive work. Setup requirements are normally minimal in assembly lines since the tasks are repetitive. Instructions are simply to repeat the previous activity. Thus, workers quickly reach effective points on the learning curve. Assembly line systems do not require large queues of work-in-process inventory.

Consequently, there is less space required, plus lower inventory holding cost and shorter throughput time (Askin and Standridge, 1993).

2.1 TERMINOLOGY USED FOR ASSEMBLY LINES

A *workstation* or *station* is a location along the line where a group of tasks is performed by operators and/or machinery. A *task* is the smallest, indivisible and rational work element of the total work content in an assembly line.

Task time is the duration of the task. It can be deterministic or stochastic. *Work content (station time)* is the actual amount of task times assigned to that station. *Cycle time* is the time between the completion times of two consecutive units. It is specified according to the required production rate to be achieved by the flow line. To reach the planned rate, the cycle time should be less than or equal to 1/planned rate.

Precedence constraints are precedence requirements that restrict the sequence in which the job can be accomplished. Job i is said to be predecessor of job j if it has to be completed before job j starts. In such a case job j is the successor of job i.

Precedence diagram is a graphical representation of the sequence of tasks defined by the precedence constraints. The processing of a task cannot start until its *immediate predecessors* have been processed. The *immediate predecessors* of a task are the tasks, which must be completed just before starting to process this task. The *immediate successors* of a task are the tasks, which can be processed just after the completion of this task. An example of precedence diagram is illustrated in Figure 2.1. The nodes in the figure represent tasks. The task times are given above each node. Arrows indicate the sequences in which the tasks must be done. According to the figure, job 1 is the immediate predecessor of jobs 2, 3 and 4, and job 5 is the immediate successor of jobs 2 and 3.

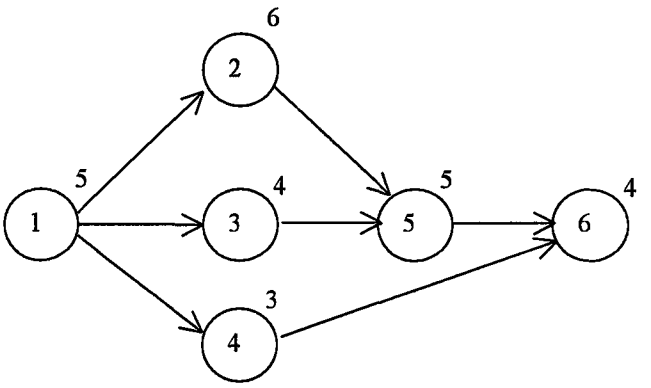


Figure 2.1 An example precedence diagram

Precedence matrix is a tabular representation of the precedence relations. It is an upper triangular matrix. Rows and columns are labeled with consecutive task numbers. If task i immediately precedes task j, 1 is placed in an entry corresponding to row i and column j. All other entries are zero. Figure 2.2 illustrates the precedence matrix of the example given in Figure 2.1.

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

Figure 2.2 Precedence matrix of the example given in Figure 2.1

The precedence structure of a line can be characterized by the *Flexibility-ratio (FR)* which is a measure of the number of feasible sequences that could be

generated from an N-task problem. A similar measure is the O*rder Strength*, which measures the volume of distinct ordering that are permitted by the specified precedence relations (Erel and Sarin, 1998). These measures can be expressed as follows:

$$\text{FR} = \frac{2*(\#of \text{ zeros in the precedence matrix})}{N(N-1)}$$

$$\text{Order Strength} = \frac{2*(\#\text{ of ones in the precedence matrix})}{N(N-1)}$$

As the FR increases, the line becomes more flexible since fewer precedence relations exist. When the order strength increases, number of precedence relations also increase. So, we can say that higher the order strength, the dependency between tasks gets higher which lead to a more strict line.

In some situations a precedence diagram cannot express all the implications of the setting up an assembly line. *Zoning restrictions* indicate which tasks must be assigned to the same workstation and which tasks must not be assigned to the same workstation.

2.2 PERFORMANCE MEASURES

There are commonly two types of performance measures used in the assembly lines. These are t*otal idle time* and b*alance delay*.

The difference between cycle time and work content is called the *idle time* of that station. *Total idle time* can be considered as a measure of design effectiveness of the line and is the sum of all station idle times.

Balance delay is a measure of the line inefficiency that results from idle time due to imperfect allocation of work among stations. It is symbolized as BD and is computed as follows:

$$BD = \frac{\text{Total idle time}}{\#\ of\ \text{stations} \times \text{cycle time}}$$

Balance delay is the ratio of total idle time to the total time spent by the product from the beginning to the end of the line. If BD equals to zero then, *perfect balance* occurs. In this situation, the work contents in all stations are exactly equal to the cycle time.

2.3 ASSEMBLY LINE BALANCING (ALB) PROBLEM

The assembly line balancing problem consists of assigning tasks to an ordered sequence of stations such that the precedence relations and zoning constraints among the tasks are satisfied and some performance measure is optimized (Erel and Sarin, 1998).

There are mainly two types of objectives used in assembly line balancing. One of them is the minimization of total idle time given a desired cycle time. The ALB problem considers this objective is called *Type I problem*. It is also important to mention that the minimization of number of stations for a given cycle time is also equivalent to this objective. The other objective is the minimization of cycle time for a fixed number of stations. The problem considers this objective is called *Type II problem*. Majority of literature is for Type I problems. However, Erel and Sarin (1998) states that there are also some other objectives like minimizing balance delay and maximizing the profit per unit of time used in literature.

2.4 PARALLELING CONCEPT IN ASSEMBLY LINE BALANCING

The main assumption in the traditional assembly line balancing problem is that the line is 'serial', i.e. there is no 'paralleling' of task. Each task is assigned to a particular workstation that can perform the task. Paralleling allows a specific task

to be performed at more than one station. The serial line assumption restricts the cycle time to at least the maximum task time, thereby limiting the production rate. However, since paralleling reduces the largest task time by performing this task in more than one station, the maximum achievable production rate increases.

There are two types of paralleling in assembly line balancing. These are *paralleling of tasks* and *paralleling of stations*.

2.4.1 Paralleling of Tasks

If the task time of some tasks in assembly line is greater than the cycle time, task paralleling can be used to balance the line. A paralleled task can be performed at more than one station. By paralleling, the effective task time is reduced by the number of times the task is replicated.

Theoretically the effective task times for a paralleled task, given by t_{ia} and t_{ib}, can be set such that $t_{ia} + t_{ib} = t_i$ where t_i is the processing time of task i. Searching for possible combinations of t_{ia} and t_{ib} makes the solution procedure very complex. Hence, in the literature, the effective task times are limited to equal allocations i.e. $t_{ia}=t_{ib}=t_i/2$.

A critical assumption in ALB is that the tasks are indivisible. It is important to note that even if a task is paralleled, the above assumption still holds. Paralleling entails performing the task at more than one station and not dividing the task.

We illustrate the task paralleling through the problem in Example 1. Precedence diagram for a 5-task assembly line is given in Figure 2.3. The production rate is such that the cycle time is set at 14 minutes.

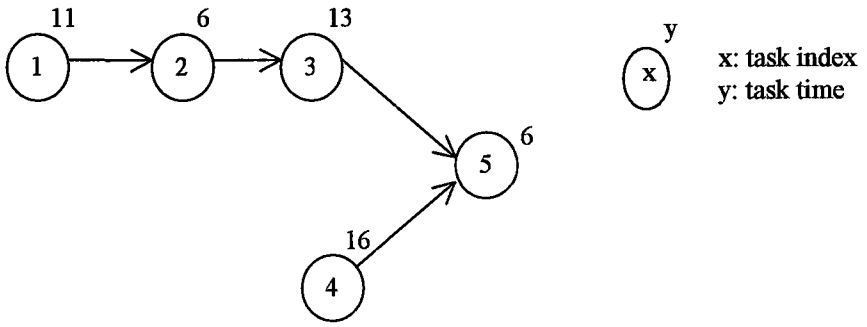


Figure 2.3 Precedence diagram for Example 1

Since the time of task 4 exceeds the cycle time, overtime is required to meet the production rate.

In the example problem, task 4 can be paralleled once, such that it can be effectively replaced by two new tasks each with a task time of 8 min. Figure 2.4 shows the precedence diagram for the situation when task 4 is paralleled.

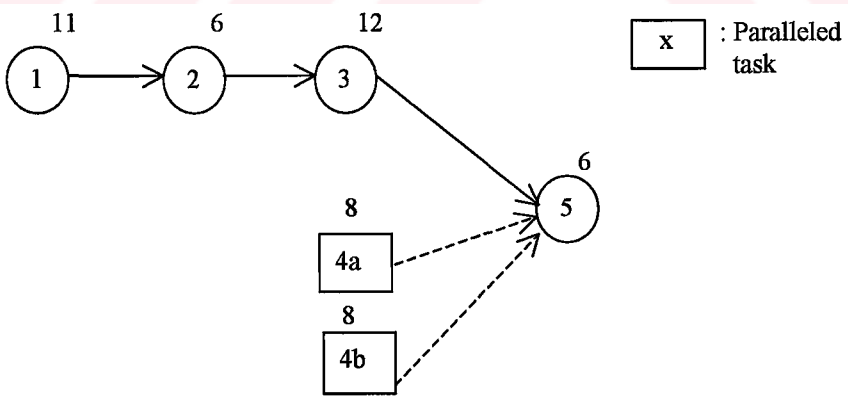


Figure 2.4 Precedence diagram with task 4 paralleled

If paralleling is not allowed, the cycle time must be increased to 16 min. The best allocation relative to the number of workstations would be:

Station #	Task assigned
1	1
2	2
3	3
4	4
5	5

If paralleling is allowed, the production rate can be met with cycle time of 14 min. In this case the line can be balanced with 4 stations as follows:

Station #	Tasks assigned
1	1
2	2,4a
3	3
4	4b,5

Decision about which tasks to be paralleled and the level of paralleling involves consideration of the cost of additional facilities for performing the tasks that are paralleled at different stations. Even though none of the task times is greater than the cycle time, paralleling of tasks can also lead to a balance with fewer stations.

2.4.2 Paralleling of Stations

Paralleling of stations implies duplication of all tasks of that station. By paralleling a station, the maximum allowed work content is increased to m*CT where m is the # of times the station is parallel and CT is the cycle time. In such a case m units will be available by the end of m*CT time units.

The main challange in paralleling of stations is to improve balance efficiency. Paralleling can reduce idle times by fitting tasks to stations more 'tightly' because the work content allowed at a paralleled station is equal to the cycle time multiplied by the number of times that station is paralleled.

As mentioned, the production rate is limited by the longest task time in traditional assembly line models. Since paralleling reduces the effect of longest task time, the production rate increases by paralleling of stations.

An additional benefit of paralleling relates to system reliability. In a serial line, the failure of any one station causes the entire line to halt; the failure of a paralleled station permits operations to continue, albeit at a reduced rate (Bard, 1989).

Although substantial benefits of station paralleling, it should not be ignored that in the paralleling of a station, an additional cost is incurred due to the duplication of all tasks in the station.

We illustrate the parallelization of stations through Example 2 taken from Bard (1989). The precedence diagram of the assembly line is given in Figure 2.5. The cycle time is 55 minutes.

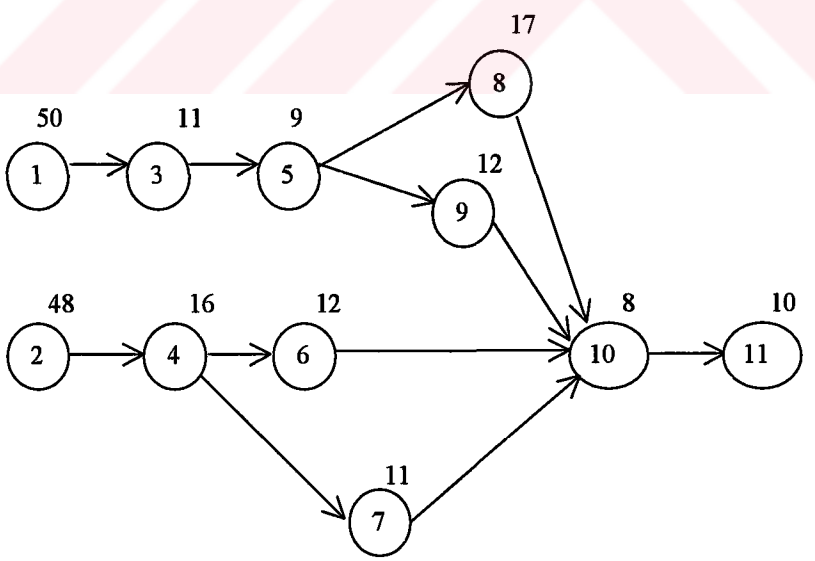


Figure 2.5 Precedence diagram for Example 2

Two balances for this line are as given by Figures 2.6 and 2.7. In Figure 2.6, the balance when paralleling is not permitted is shown whereas in Figure 2.7, the balance is shown when paralleling of stations is permitted.

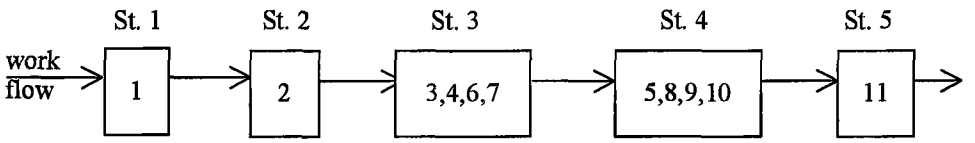


Figure 2.6 Balance with serial stations

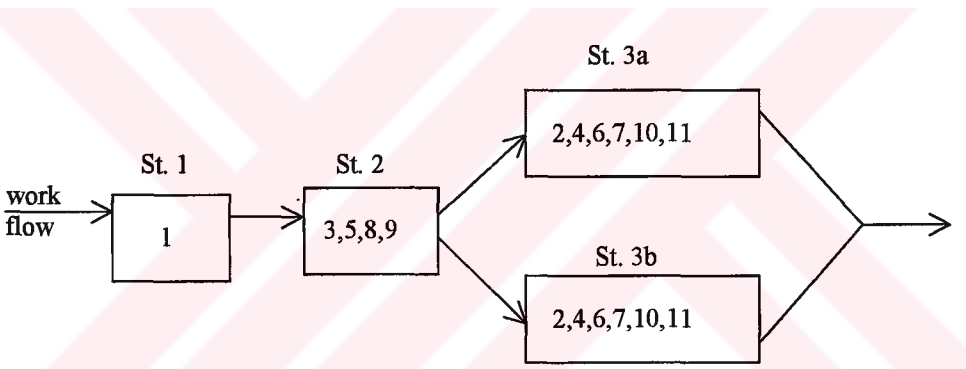


Figure 2.7 Balance with parallel stations

When paralleling of stations is not permitted, the line is balanced with 5 stations. The efficiency of the line shown in Figure 2.6 is calculated as follows:

Total idle time = 5+7+5+9+45 $\Rightarrow$ Total idle time = 71 min.

BD = 71/5*55 $\Rightarrow$ BD = 0.26

Efficiency of the line = 1-0.26 $\Rightarrow$ Efficiency of the line = 74%

When paralleling of stations is permitted, the line is balanced with 4 stations. The efficiency of the line shown in Figure 2.7 is 93%. Note that by paralleling of

stations, a more efficient line with less number of stations is obtained. But it must also be considered that in the paralleling of a station, the additional cost incurred is the cost of duplicating all production facilities required at that station. Hence, this cost becomes an important performance measure to decide if the paralleled line is desirable.

CHAPTER 3

ASSEMBLY LINE BALANCING LITERATURE

In ALB literature, the problems are classified into four categories as shown in Figure 3.1: Single-Model Deterministic (SMD), Single-Model Stochastic (SMS), Multi/Mixed Model Deterministic (MMD) and Multi/Mixed Model Stochastic (MMS).

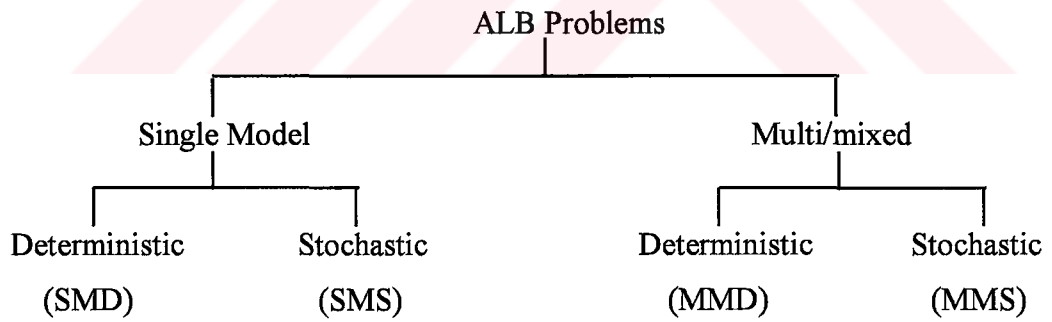


Figure 3.1 Classification of the ALB problems

The SMD problem applies to single-model assembly lines where the task performance times are known constants. This is the simplest form of the ALB problem. The SMS category introduces the concept of task time variability. This version represents the manual assembly lines more realistically where task performance times are seldom constants. The variability of task times can be rather

large relative to their means. This is especially true for tasks that are complex and require high levels of skill and concentration (Erel and Sarin, 1998). Ghosh and Gagnon (1989) state that with the introduction of stochastic task times many other issues become relevant, such as station times exceeding the cycle time, thereby the production of defective or unfinished parts, pacing effects on workers' operation times, station lengths, size and location of inventory buffers, launch rates, and allocation of line imbalances.

In MMD problem more than one model on a single line with deterministic task times are produced. Hence model selection, model sequencing and launching rate(s), and model lot sizes become more critical issues. The MMS problem differs from MMD in that stochastic times are allowed. All factors arising from stochasticity that are relevant in the SMS problem also become relevant in MMS. However, these issues become more complex for the MMS problem because factors such as learning effect, worker skill level, job design and worker task time variability become more difficult to analyse because the line is frequently rebalanced for each model assembled (Ghosh and Gagnon, 1989).

In this study, we are interested in the SMD assembly line balancing problem and we exclude any stochastic ALB problems in our literature review.

3.1 OVERVIEW OF ALB PROCEDURES/ALGORITHMS

ALB problems have a great computational complexity. Even Type I problem (also called SALB-1; Simple Assembly Line Balancing Type I) is NP-hard. Although the formulation of the problem is not difficult, finding feasible task sequences that minimize the number of stations requires enormous effort. If the precedence relations are disregarded, the problem looks like a sequencing problem and N! solutions exist for N tasks. Since there are extremely large number of feasible solutions, an enormous effort is required to find the optimum sequence among these feasible solutions. However, the precedence restrictions and cycle time constraints reduce the number of feasible solutions. It is stated in Erel and Sarin

(1998) that if there are r precedence relations among N tasks, then there are roughly N!/2r distinct sequences.

Since the first formulation of assembly line balancing problem by Salveson in 1955, much progress has been made in this area and many techniques have been developed. The most recent papers that review the work on this subject are due to Baybars (1986), Ghosh and Gagnon (1989), Erel and Sarin (1998). Baybars (1986) discusses the development of the simple assembly line balancing problem as well as its modifications and generalizations over time. He provides comparison and analysis of exact and heuristic algorithms for the SMD version of the problem. Ghosh and Gagnon (1989) report the results of a comprehensive review and analysis of the ALB literature. Quantitative developments and qualitative issues are addressed at both the strategic and tactical levels. They organize numerous quantitative and qualitative factors which could impact the design, balancing and scheduling of assembly systems into an eight-level hierarchical, factor/decision taxonomy. This taxonomy is used to access the progress in assembly system design and operation. Erel and Sarin (1998) critically examine the heuristic procedures in ALB literature and summarize them to provide a state-of-the-art survey. They also present the evaluation of the procedures and some topics in need of further research.

The solution procedures in ALB can be classified as exact methods and heuristic methods. The exact methods include Integer Programming, Dynamic Programming, Branch & Bound Algorithm (optimal tree search algorithms), Shortest Path or Maximal Path algorithms.

The most widely used heuristic method is the priority ranking approach, where the tasks are ranked according to some criteria or priority rule like longest task time and largest positional weight and then assigned to stations. Tree search or enumerative methods, which are essentially branch and bound methods, except that the branching, fathoming and termination rules are heuristic, come next. Trade and transfer methods first achieve a balance by various methods and then

interchange tasks between stations to achieve a better balance. Random sampling methods, known as random generation techniques, assign tasks to stations randomly or randomly select a rule (from a set of rules) for assigning a task to a station (Ghosh and Gragnon, 1989).

Ghosh and Gragnon (1989) also state that, although integer and dynamic programming formulations have been extensively used to express the ALB problem, neither can be efficiently solved by available techniques. Due to this, enumeration techniques, branch-and-bound methods and dynamic programming algorithms have gained widespread use for solving the integer programming formulations.

3.2 SINGLE MODEL DETERMINISTIC ASSEMBLY LINE BALANCING

The procedures developed related with ALB have been mostly in this category. Salveson (1955) was the first to publish the analytical statement of the Type I problem. Thangavelu and Shetty (1971), Patterson and Albracht (1975), Talbot and Patterson (1984) were the others who formulated the Type I problem as integer programming formulations.

Although integer programming based approaches may be impractical for especially big problems, it would be useful to give the general structure of the formulation. We give the mathematical formulation of the Type I problem taken from Sprecher (1999). We also add zoning restrictions to this formulation to present a more general one.

In the formulation, an assembly line with J tasks is considered. The precedence relations are described by the sets of predecessors P_j of the tasks $j=1,\ldots, J$, indicating that a task j can not be processed before all tasks h, $h \in P_j$, are completed. The processing times of tasks are deterministic. In the precedence diagram it is assumed that tasks are numerically labelled, i.e. a task always has a higher number than all its predecessors. When the precedence diagram is thought as a network,

without loss of generality, the network has exactly one source j=1 and one sink j=J with a zero duration. The tasks have to be assigned to linearly ordered stations. The notation used in the model is given below.

Indices

j	Job index
v	Station number index

Parameters

J	Number of tasks
t_j	Processing time of task j
c	Cycle time
V	Upper bound on the number of stations
P_j	Set of immediate predecessors of task j
M	Set of pairs of tasks (a, b) that must be processed in the same station
N	Set of pairs of tasks (a, b) that must not be processed in the same station

Decision variable

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

The mathematical formulation of the SALB-1 is as follows:

$$\text{Minimize } Z(x)= \sum_{v=1}^{V} v.x_{jv} \qquad (1)$$

s.t.

$$\sum_{v=1}^{V} x_{jv} = 1 \qquad j=1,\ldots, J \qquad (2)$$

$$\sum_{j=1}^{J} t_j.x_{jv} \le c \qquad v=1,\ldots,V \qquad (3)$$

$$\sum_{v=1}^{V} v.x_{hv} \le \sum_{v=1}^{V} v.x_{jv} \qquad j=1,\ldots,J,\ h\in P_j \qquad (4)$$

$$\sum_{v=1}^{V} v.x_{av} - \sum_{v=1}^{V} v.x_{bv} = 0 \qquad (a, b) \in M \qquad (5)$$

$$x_{av} + x_{bv} \le 1 \qquad (a, b) \in N,\ \forall v \qquad (6)$$

$$x_{jv} \in \{0,1\} \qquad j=1,\ldots,J,\ v=1,\ldots,V \qquad (7)$$

Since we have only one finishing task, the objective (1) realizes the minimization of the number of stations. Constraint set (2) ensures that each task is assigned to exactly one station. Constraint set (3) guarantees that the sum of the processing times of tasks assigned to one station do not exceed the cycle time c. The precedence relations are taken into account by (4). Zoning restrictions are taken into consideration by constraints (5) and (6). Constraint set (5) ensures that the tasks are processed in the same station while constraint set (6) ensures the tasks are processed in different stations.

Most procedures optimally solving Type I problem are based on branch-and-bound procedures. Jackson (1956) was the first who constructed the branch and bound algorithm. His method explicitly examines all possible assignments up to the last station before examining any assignments to the last station. He showed that an optimal solution exists in an enumeration tree whose arcs represent only maximal stations, where a station is maximal if there is not any unassigned task that can be

added into this station feasibly without exceeding the cycle time or violating the precedence constraints.

Although there are several branch and bound procedures developed in the literature, the most widely known and used branch and bound algorithms in comparison studies include Johnson (1988), Hoffmann (1992), Sholl and Klein (1997), and Sprecher (1999).

Johnson (1988) has developed a laser type, depth-first, branch and bound algorithm called FABLE (Fast Algorithm for Balancing Lines Effectively). In FABLE, tasks are renumbered prior to tree generation. A task is renumbered only after all its preceding tasks are renumbered. Therefore, task renumbering starts with a task which has no preceding tasks. Ties are broken in favor of tasks with longer durations. Second level ties are broken in favor of the task with the greatest number of immediate successor tasks. Forming the tree one branch at a time is termed laser search. Hence, a feasible solution is found in the first complete branch of the tree. Because tasks are always assigned to the earliest station available (task-oriented branching), the stations are filled in forward direction. It uses four different rules for node fathoming. Although it is an effective algorithm, it has some limitations. Some of the fathoming rules cannot be applied if certain tasks must be restricted to specified stations. Sholl and Klein (1999) state that although the logical tests and bounds used greatly reduce the size of the tree, the algorithm may be trapped within its rigid enumeration scheme so that the optimal solution is not reached within a reasonable computation time available.

Hoffmann (1992) developed a branch and bound algorithm called EUREKA. It makes use of a simple bounding rule, based on the theoretical minimum slack. The procedure branches in a station-oriented manner by assigning a complete load to the earliest unloaded station at a time. It first computes the theoretical minimum number of stations and then starts searching for an optimal solution in forward planning. If no optimal solution is found within the allowed time, it starts search for optimal solution in backward planning. Again if no optimal solution is found

during backward searching, then using the heuristic method of Hoffmann (1963) a feasible solution is found. It is stated in Scholl and Klein (1999) that if the optimal number of stations is significantly larger than the initial lower bound, powerful dominance rules and lower bounds are needed to reach an optimal solution in reasonable computation times. They state that the additional backward planning of EUREKA sometimes quickly finds optimal solutions where the forward procedure fails.

Scholl and Klein (1997) developed a bidirectional branch and algorithm called SALOME (Simple Assembly Line Balancing Optimization Method). It has station-oriented branching procedure. The tasks are renumbered as in FABLE. Since it is a bidirectional procedure, it makes use of both the original and the reversed precedence graph. It has also some logical tests like maximum load rule. Scholl and Klein (1999) state that SALOME reaches to good and often optimal solutions very quickly due to its bidirectional enumeration scheme. According their computational results, SALOME clearly outperforms FABLE and EUREKA for all performance measures concerned. The performance measures that they consider are number of instances for which an optimal solution is found, average and maximum relative deviation from optimality in percentage, and average computation time in seconds.

Sprecher (1999) presents a branch and bound algorithm called AGSA (Adapted General Sequencing Algorithm). The algorithm relies on the precedence tree guided enumeration scheme introduced for dealing with a broad class of resource-constrained project scheduling problems (RCPSP). The problem can be considered as a RCPSP with a single resource with time-varying availabilities and activities using one unit each period they are in process. Thus, the problem can be solved by any RCPSP algorithm capable of dealing with time varying availabilities. Before the enumeration starts, the tasks are relabelled with respect to the maximum duration rule. In his study, he also compares AGSA with FABLE, EUREKA and SALOME. Computational results show that AGSA can compete quite well with these procedures.

The mathematical formulation for Type II problem is the same as the Type I problem except the objective function. Here, our objective function is for minimizing the cycle time for a fixed number of stations.

In the literature, there are not many procedures developed for solving the Type II problem. This is because one can solve any Type II problem by successively applying Type I solution procedures with various trial cycle times. The most recently developed procedures for solving Type II problems are Klein and Scholl (1996) and Uğurdağ et al. (1997).

Klein and Scholl (1996) developed a branch and bound algorithm called SALOME-2 which is the adaptation of their SALOME to the Type II problem. The procedure uses a new enumeration technique, namely the local lower bound method, which is complemented by a number of bounding and dominance rules. Computational results indicate that the new procedure is very efficient.

Uğurdağ et al. (1997) provide a two stage heuristic procedure, which is based on an integer programming formulation of the problem. It is a simplex-like procedure which, while attempting to minimize the cycle time, also smooths out the workload among the workstations. The first stage of the procedure finds an initial solution to the problem. The second stage of the algorithm improves the initial solution using a simplex-like procedure. Computational results show that their procedure performs well.

As exact algorithms become intractable with increasing problem size, considerable effort has been spent in development of heuristics. These methods can give a good solution in a short time even for large problems.

In the literature there are many heuristic procedures developed for solving the Type I problem. In this study only six most popular ALB heuristics by Helgeson and Birnie (1961), Kilbridge and Wester (1961), Moodie and Young (1965), Arcus (1966), Hackman et al. (1989) and Hoffmann (1963) are discussed. Erel and Sarin

(1998) give a detailed review of the heuristics in the literature.

Helgeson and Birnie (1961) propose the well-known and popular Ranked Positional Weight Technique (RPWT). In this technique a positional weight for each task, which is the sum of its task time and the task times of all other successor tasks, are calculated. Tasks are ranked in descending order based on their positional weights and assigned to stations in that order. If a task takes longer than the cycle time remaining in the station or would violate the precedence relations, then the next task is considered. If no further task can be assigned to a station, the next station is opened. The technique does not guarantee a very good solution, but by using different cycle times several alternative balances can be found.

Kilbridge and Wester (1961) propose a technique developed primarily to balance lines without the aid of a computer. The main feature of the technique is to group tasks into columns in the precedence diagram where tasks are placed as far to the left as possible without violating the precedence relations. It is a simple, powerful technique, especially for long cycle times where one station crosses several columns. On the other hand, with short cycle times one column may require two or more stations, and a fair amount of adjustment is necessary (Erel and Sarin, 1998).

Moodie and Young (1965) developed a two-phase procedure. In phase 1, the tasks are assigned to stations by the largest candidate rule (longest task time) without violating the precedence relations. In the second phase, by the mechanism of trades and transfers of tasks, an attempt is made to reduce and distribute the idle time equally among the all stations.

Arcus (1966) developed a technique called COMSOAL (Computer Method of Sequencing Operations for Assembly Lines). A task is said to be fittable to the current station if its task time is not greater than the remaining station time and all the predecessor tasks of that task have been assigned. The technique selects a fitable task randomly and assigns it to the station. This procedure goes on until there are no unassigned tasks.

Hackman et al. (1989) developed a branch and bound procedure. The nodes are fathomed according to several heuristic rules and this reduces the size of the tree.

Hoffmann (1963) proposed an enumeration method using a triangular precedence matrix. The method generates all feasible station assignments that do not exceed the cycle time. The first station is obtained by selecting the feasible subset of tasks that minimize station idle time. The second station is obtained by selecting the feasible subset of tasks that leaves the least station idle time from the remaining tasks. This procedure is repeated until all tasks are assigned.

Ponnambalam et al. (1999) conducted a comparative evaluation study. In their study, they use some heuristic including Helgeson and Birnie (1961), Kilbridge and Wester (1961), Moodie and Young (1965), Hackman et al. (1989), and Hoffmann (1963). All the methods are compared for four performance measures. The performance measures used are number of excess stations, line efficiency, smoothness index and CPU time. The results show that Hoffmann (1963)' s enumeration procedure performs best. However, Hoffmann's procedure enumerates all the feasible alternative sets of tasks at the stations therefore its execution time is the longest.

3.3 MULTI/MIXED MODEL DETERMINISTIC ASSEMBLY LINE BALANCING

Multi Model ALB and Mixed Model ALB differ in the lot sizes. On a mixed model line the lot sizes equal one. Because of this, the change-over costs and change-over times should be negligible. But in a Multi Model ALB problem, the lot sizes are greater than one and therefore the change-over costs can not be assumed negligible. So, the objective of the Multi Model ALB problem is to minimize the production costs which include change-over costs.

The solution procedures developed for solving Multi/Mixed Model ALB problems are fewer compared to the Single Model ALB procedures. As Fokkert and de Kok

(1997) state, most of the articles in this area date back to 1980s. However, because of the evolution of change-over problems in mass assembly, they foresee a renewed interest in the Multi/Mixed Model ALB.

The Single Model ALB and the Mixed Model ALB differ in the precedence constraints and task times. In Mixed Model ALB, every model has its own precedence diagram and a balance satisfies any of these orderings. However, only one precedence diagram and identical task times are given in Single Model ALB.

Thomopoulos (1967) proposed a procedure for Mixed Model ALB. He used a combined precedence diagram, a combination of different models into a single precedence diagram, and then he used the heuristic of Kilbridge and Wester (1961) to minimize the number of stations in his procedure. It must be noted that in Mixed Model ALB, the cycle time is defined according to the duration of a shift.

Macaskill (1972) also proposes a combined precedence diagram based procedure for Mixed Model ALB problems. He used the heuristic of Helgeson and Birnie (1961) to minimize the number of stations.

Gökçen and Erel (1997) presented an optimum seeking (exact) algorithm for Mixed Model ALB. They developed a binary integer programming model that utilizes a combined precedence diagram for different models in the problem. They have reported optimal solutions of problems with up to 40 tasks in the combined diagram.

Wild (1972) suggests that when the lot sizes are small and when it is important that every repetition of a task is carried out by the same station, a balancing method for mixed model lines can also be used to balance a multi model line. He also proposed to balance a multi model line with large lot sizes by successive application of single model ALB procedures. This procedure balances the line for the model that has the largest frequency. The remaining models are balanced in

such a way that similar tasks are assigned to the same station in order to minimize the change-over activities.

Chakravarty and Shtub (1985) proposed a heuristic procedure for Multi Model ALB, which considers the labour, set-up, and the inventory costs. In the procedure, by placing buffers between two adjacent stations, the batch sizes are allowed to vary between stations.

Berger et al. (1992) proposes a branch and bound algorithm with a truncated search for Multi Model ALB in which the common tasks of the models are performed first. The production process of the models starts diverging after the common tasks are performed. This algorithm provides an exact solution if its full version is used.

3.4 ASSEMBLY LINE BALANCING WITH PARALLELING

Almost all the procedures developed for assembly line balancing assume that the longest task time in a balancing problem is less than the predetermined cycle time. This type of situation is usually found in sophisticated and/or smaller product industries, especially in electrical/electronics industries. All these techniques fail to solve the balancing problems when at least one task time in the system is longer than the cycle time (Sarker and Shanthikumar, 1983).

In the literature, the studies on balancing assembly lines by using the paralleling concept are few. Buxey (1974) explains how to balance the assembly lines with parallel operation both for the stations themselves and the line as a whole. He also explains some assembly line design issues. He uses the Ranked Positional Weight Technique and the Random Generation method as a basis for balancing the line by incorparating the concept of multiple (parallel) stations to both heuristics.

Pinto et al. (1975) develope a branch and bound algorithm to balance the assembly lines when paralleling of tasks is allowed. They state that, although paralleling

involves consideration of the cost of additional facilities for performing the tasks that are paralleled at different stations, the paralleled line may in many cases be a less costly method of increasing production than the other alternatives such as the use of overtime, another assembly line, subcontracting and buffer stocks. When balancing assembly lines with paralleling of tasks, the main problem is selecting the tasks to be paralleled in such a way that the total relevant costs are minimized. These costs are the regular time labour cost, the cost of duplicating facilities for performing paralleled tasks, and any overtime cost yet required for meeting the desired production rate. The branch and bound algorithm proceeds by partitioning the set of all combinations 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. Initially, all the tasks are in an undecided state. A 'full' combination is achieved by explicitly fixing all tasks either to be paralleled or not paralleled. The procedure starts by considering all tasks as not paralleled and this solution is used as an upper bound. If the lower bound on the total cost equals to the upper bound, the optimal solution is reached, algorithm stops. If not, the algorithm continues with branching until it finds an optimal solution. In this algorithm, the fixed costs are first converted into equivalent manpower to make the cost of fixed facilities and the labour cost comparable. Thus, the objective is minimization of the total equivalent manpower plus the number of stations.

Pinto et al. (1981) developed a second branch and bound algorithm to balance the assembly lines with paralleling of stations. The algorithm is very similar to Pinto et al. (1975)' s algorithm. In the objective function, both the labour cost and the cost of additional production facilities are considered when the station is paralleled. Again, the problem is converted from one of cost minimization to minimizing the equivalent manpower plus the number of stations. The steps of the algorithm is the same as in Pinto et al. (1975). The only difference being, a partial combination is made up of stations that can be classified as station 'fixed paralleled', station 'fixed not paralleled' and station yet 'undecided'. They also propose a heuristic solution procedure derived from the branch and bound method.

Sarker and Shanthikumar (1983) develop a generalized heuristic to balance an assembly line where the task times may be shorter or larger than the specified cycle time. The approach is composed of two phases. The first phase deals with assigning the tasks to different stations according to the longest task time rule, and the second phase deals with reducing the balance loss of the line by trades and transfers. The paralleling, which is used either to increase the efficiency of the line and/or balance the line when the cycle time is less than at least one task time, is adopted during the first phase.

Bard (1989) proposed a dynamic programming algorithm for solving an assembly line balancing problem with parallel workstations. The objective is again the cost minimization. The main difference of this algorithm from Pinto et al. (1975) and Pinto et al. (1981) is that, it includes the cost of duplicating both tasks and stations. A second distinguishing feature of the algorithm is that it takes the unproductive or dead time at a station into account where dead time is defined as the fixed set-up time that might accompany the movement of a conveyor, or the time it takes for a robot to move in on the workpiece.

Gökçen (1997) proposes a heuristic method that provides paralleling of the stations after balancing of the line. This method has the potential of improving the balance efficiency of the line by reducing the station idle times and increasing the production rate. In the method, paralleling concept is considered after a single line is found. The stations with higher idle times are paralleled and then grouped with their adjacent stations to reduce the idle time. There are two decisions related with paralleling. First decision is whether paralleling is required or not. If the number of stations after balancing equals to the theoretical minimum number of stations, then there is no need for paralleling. However, if the number of stations after the balance is greater than the theoretical minimum number of stations and the total idle time is greater than or equal to the cycle time, then paralleling is considered. The second decision is about the choice of stations to be paralleled. This decision is based on the comparison of idle time at each station with the limit on the

maximum station idle time allowed by the management. Every station whose idle time is greater than this value is paralleled. The paralleled stations are then grouped with adjacent stations without exceeding the cycle time.

Askin and Zhou (1997) develop a heuristic for mixed model assembly line balancing with paralleling. To our knowledge, theirs is the first study considering task-dependent equipment cost and its effect on the station paralleling decision in Mixed Model Assembly Line Balancing literature. They state that creating parallel, identical workstations ameliorates the worker idle time problem by increasing the effective cycle time for each stage in the serial production process but at the expense of additional capital cost. As they mention in repetitive manufacturing industries, such as automobiles, watches and clocks, refrigerators etc., equipment duplication cost can be significant and should be taken into account. From this point of view, their algorithm is important in that it considers task-dependent equipment cost and station paralleling *simultaneously*. The heuristic allows a station to be paralleled arbitrary number of times and uses a threshold to take into account the utilization requirement. The decisions on paralleling of workstations are made not only based on the task times, but also on the comparison of incremental equipment/tooling cost with the cost of unutilized workstation capacity. The heuristic can quickly find good solutions to large problems. As their study is very closely related to our study, more information about their algorithm and its modification according to our purposes are given in Chapter 4.

McMullen and Frazier (1998) propose a Simulated Annealing based technique to address the assembly line balancing problem with multiple objectives when paralleling of workstations is allowed. In their study, they relax several common assumptions for line balancing problems. First, task times are assumed to be stochastic. Second, the assumption of a single objective is relaxed. Two primary objectives are emphasized in the study, the extent to which the desired cycle time is achieved and the total design cost of the production line (equipment cost and labour cost). Third, they relax the assumption of one worker per station and allow

using of parallel workstations. Finally, the assumption of a single product type is relaxed. The algorithm can be used both in Single Model ALB and in Mixed Model ALB. The experimental results show that the approach yields significantly better solutions in cycle time performance but average solutions in cost performance.

It would be useful to mention that Pinto et al. (1975), Pinto et al. (1981), and Bard (1989) allow 2-level paralleling. That is, only single duplications of tasks or stations are considered. Other researchers allow more than 2-level paralleling in their procedures.

As seen, most of the procedures developed for balancing assembly lines with paralleling consider cost minimization. Askin and Zhou (1997) stress the importance of considering equipment cost when paralleling. However, their algorithm is an heuristic algorithm and does not guarantee optimality. Among the studies on SMD ALB, there is no exact procedure that considers task dependent equipment cost and station paralleling at arbitrary number of times. We propose a branch and bound algorithm for SMD ALB which considers task dependent equipment cost and station paralleling *simultaneously*. This branch and bound algorithm is an exact procedure under the assumption that a station is never closed as long as there is a fitable task and it allows more than 2-level paralleling.

CHAPTER 4

BRANCH AND BOUND ALGORITHM

In this chapter, we describe our approach to solve SMD ALB problem with station paralleling. We first give the mathematical representation of the problem together with its underlying assumptions. Our branch and bound approach together with the upper and lower bounds employed is explained next.

4.1 MATHEMATICAL FORMULATION

'Stages' are the major segments of a line. Each segment contains a part of the total work content or total task. Each segment may require different number of operators or machines to balance the output rate of one segment to the segment immediately following it. The operators or machines at each stage are known as the 'stations'. The number of stations at each stage represents the order of paralleling.

We study the SMD ALB problems with the following assumptions:

- A single product is assembled on the line. The processing times of tasks are deterministic.
- Station paralleling is allowed.
- A station is never closed if there exists any fitable task.
- Each task requires a specified type of equipment to be performed. An equipment can be a machine or an instrument or a tool that is required to process the task.

- Each type of equipment is associated with a specific cost. This cost can represent the purchasing and operational cost of using the equipment.
- When station paralleling occurs, the equipment(s) in that station is(are) also duplicated.
- The cycle time to satisfy the demand is known and fixed.
- The precedence restrictions that give information about the relative processing order of the tasks are known.
- A task can be performed at any station provided that the equipment for this task is available in that station and the precedence restrictions are satisfied.
- There is a fixed cost of opening a workstation. We assume this cost includes the labor and overhead costs.

Our problem is to assign each task to a stage in the serial production system and determine the number of identical parallel stations at each stage. A workstation can be paralleled more than once. Our model is a slight modification of the model proposed in Askin and Zhou (1997). Askin and Zhou (1997) give the formulation for mixed model whereas we consider single models. The following notation is used in the model.

Indices

j	Task number
ℓ	Type of equipment
k	Stage number

Parameters

N	Number of tasks
c	Cycle time (units)
L	Fixed cost to open a station
A_ℓ	Amortised unit cost for equipment type ℓ
t_j	Processing time of task j
IP	Set of task pairs (u, v) such that task u must precede task v
D	Number of equipment types
$\ell(j)$	Equipment type required by task j

Decision Variables

y_k Number of parallel stations at stage k

$w_{\ell k}$ $\begin{cases} 1, \text{ if equipment type } \ell \text{ is required at stage k} \\ 0, \text{otherwise} \end{cases}$

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

The mathematical formulation of the problem that allows station-paralleling and considers equipment duplication cost is given below:

$$\text{Minimize } Z = \sum_{k=1}^{N} y_k . L + \sum_{k=1}^{N} y_k \sum_{\ell=1}^{D} w_{\ell k} . A_\ell \qquad (1)$$

subject to:

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

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

$$\sum_{j=1}^{N} t_j . x_{jk} \le c . y_k \qquad \forall k \qquad (4)$$

$$x_{vk} \le \sum_{h=1}^{k} x_{uh} \qquad \forall (u, v) \in IP \qquad (5)$$

$$x_{jk} \in \{0, 1\} \qquad \forall j, k \qquad (6)$$

$$w_{\ell(j)k} \in \{0, 1\} \qquad \forall j, k \qquad (7)$$

$$y_k \ge 0 \text{ and integer} \qquad \forall k \qquad (8)$$

The objective (1) is the minimization of total fixed cost for operating stations and total equipment cost. Constraint set (2) guarantees that each task is assigned to exactly one stage where N is an upper bound on the number of stages. Constraint set (3) ensures that a task cannot be assigned to a stage unless its corresponding equipment is also assigned to that stage. Constraint set (4) ensures that the sum of the processing times of the tasks assigned to one stage cannot exceed the cycle time. The precedence restrictions are taken into account by constraint set (5) such

that if task v is assigned to stage k, all its predecessors should have been assigned to stage k or one of the prior stages. Constraint sets (6) and (7) define x_{jk}' s and $w_{\ell(j)k}$' s as 0-1 variables, respectively. Constraint (8) ensures that y_k' s are non-negative integers. It is important to state that the cost structure effects the type of optimization. When equipment costs, A_ℓ s, are close to station opening cost, L, the procedure tries to minimize both costs at the same time, thereby leading to a *simultaneous optimization* problem. When A_ℓ s are much lower relative to L, the procedure gives priority to the minimization of total station opening cost and then tries to minimize total equipment cost. In such a case, *hierarchical optimization* problem occurs and the minimization of the number of stations becomes first priority objective. Among the optimal solutions relative to first priority objective, we select the one that minimizes the second priority objective, that is minimization of total equipment cost.

4.2 SOLUTION PROCEDURE

A branch and bound algorithm is developed for minimizing the total cost which considers task dependent equipment cost and station paralleling simultaneously. The solution is formed by *forward planning.* According to this, the algorithm first forms the stations of the first stage. Since one task per node of the branch and bound tree is assigned, the branching scheme is *task-oriented.* The algorithm uses *depth-first search* with complete branching (left-most strategy). According to this strategy, the nodes are generated and the node with the best value for the total cost is selected for branching. The remaining nodes are sorted in increasing order of total cost and stored in a local candidate list. At each revisit of the current node, the next node is removed from the list and selected for branching.

Throughout the algorithm, the stages and the stations in stages are opened sequentially. Tasks are assigned to stages with their equipment. If the equipment required by any task is available at a stage, then there is no need to assign the same

equipment again when assigning this task to this stage. Any task j is fitable to the station at the last opened stage if:

i. The predecessors of task j have been assigned
ii. The time to perform task j is not greater than the idle time of the station

Each node at level r represents a partial solution with r tasks assigned. The branch originating from a node represents the addition of a task to the current partial sequence. The addition can be of three types:

- *Type I additions:* An assignment to the current stage.
- *Type II additions:* An addition of a new workstation to the current stage, i.e. increasing the level of paralleling.
- *Type III additions:* An assignment to the next stage, i.e. closing the current stage.

We always follow Type I additions if there is a fitable task to the current stage. In case there is no fitable task, Type II and Type III additions are considered simultaneously. For Type II additions we reconsider the fitable tasks and add $\left(\left\lceil \frac{\text{Fultim}_k + t_j}{c} \right\rceil - y_k\right)$ workstations to the current stage k thereby increasing the level of paralleling by $\left(\left\lceil \frac{\text{Fultim}_k + t_j}{c} \right\rceil - y_k\right)$, where Fultim_k is the total time of tasks at each station at stage k. Type III additions refer to closing the current workstation and starting with $\left\lceil \frac{t_j}{c} \right\rceil$ workstations for the node that considers the addition of task j.

An accumulated cost AC_x, representing the total cost occurred upon thus far and the cost of adding this task to the current or a new stage is assigned to each node x in the tree. At each level, the nodes are sorted in increasing order of AC_x. Then, lower bounds are calculated for each node. If lower bound on total cost is greater

than or equal to the best known upper bound on total cost, this node is fathomed. If all nodes are fathomed at any level, we backtrack to the previous level. Otherwise, the node with minimum AC_x value is selected for branching.

We find an initial upper bound by the heuristic procedure of Askin and Zhou (1997) discussed in Subsection 4.2.1, and update the upper bound whenever a complete solution with smaller total cost is reached. We terminate our branch and bound algorithm whenever all nodes in the tree are searched explicitly or implicitly, i.e. the lower bounds are found to be no less than the best known upper bound. The lower bounds are explained in Subsection 4.2.2.

4.2.1 Upper Bounds

We modified the heuristic of Askin and Zhou (1997) for SMD ALB problems and used this modified algorithm to obtain an initial upper bound. As a difference, the modified algorithm considers only one type of model in the assembly line whereas the heuristic considers more than one model type. The general idea of paralleling and task assignment is the same as the algorithm of Askin and Zhou (1997).

Paralleling of stations is considered in two cases. The first case occurs when the processing time of any task is greater than the cycle time. The station must be paralleled, as it is not possible to finish the task in one cycle. In such a case, we compete for a work content of y_k*c once we have y_k parallel stations. The second case occurs when there is no task that could fit into the current station, but there is considerable idle time left in the station. In this case paralleling of stations may be advantageous. If we close the station and start a new stage, this can yield excessive idle time and larger number of stations in optimal solution. On the other hand, if we parallel the stations, the equipment duplication cost depending on the assigned tasks can be very high. So, there is a trade-off between a fixed workstation cost and paralleling. For catch this trade-off, we adopt the following strategy:

We denote the utilization of the current station at stage k as μ_k and let η be a threshold value for minimum station utilization required closing the station specified by the decision maker. When $\mu_k < \eta$, the decision to create a parallel station will be based on incremental equipment cost resulting from adding a parallel station to stage k. This cost is the sum of the cost of all duplicated equipment. If we denote the incremental equipment cost as ΔT_k, then $\Delta T_k = \sum_{\ell \in V_k} A_\ell$ where V_k is the set of tooling required to perform the tasks assigned to stage k.

If there are y_k parallel workstations at stage k, the penalty cost of idle time for closing the current workstation can be calculated by the following expression;

$$\Delta P_k = L * y_k * UC_k$$

where ΔP_k is the penalty cost of closing that workstation and UC_k is the unutilized capacity for each station at stage k. If the penalty cost of closing the current station is less than the incremental tooling cost caused by paralleling, then the decision is to "close the current station and open a new one". Otherwise, the decision is to "parallel the current station". To fasten the decision for paralleling, let μ_{max} be the maximum acceptable station utilization to parallel the station directly. If $\mu_k \leq \mu_{max}$, then the current station is automatically paralleled.

The task assignment procedure begins initially with ranking the tasks in decreasing order of their positional weights as in RPWT proposed by Helgeson and Birnie (1961). At each step the first fitable task is added to the current stage. A task is fitable if all its predecessors have been assigned and its task time is no greater than the current idle time. If no fitable task exists, either a new station in a new stage is opened or the current station is paralleled. It must be noted that when a parallel station is added to stage k, the idle time of each station at stage k (T_k^a) is $c - \left(\sum_{j \in S_k} t_j \right) \Big/ y_k$ where S_k is the set of tasks assigned to stage k. We give the flowchart of the algorithm in Figure 4.1.

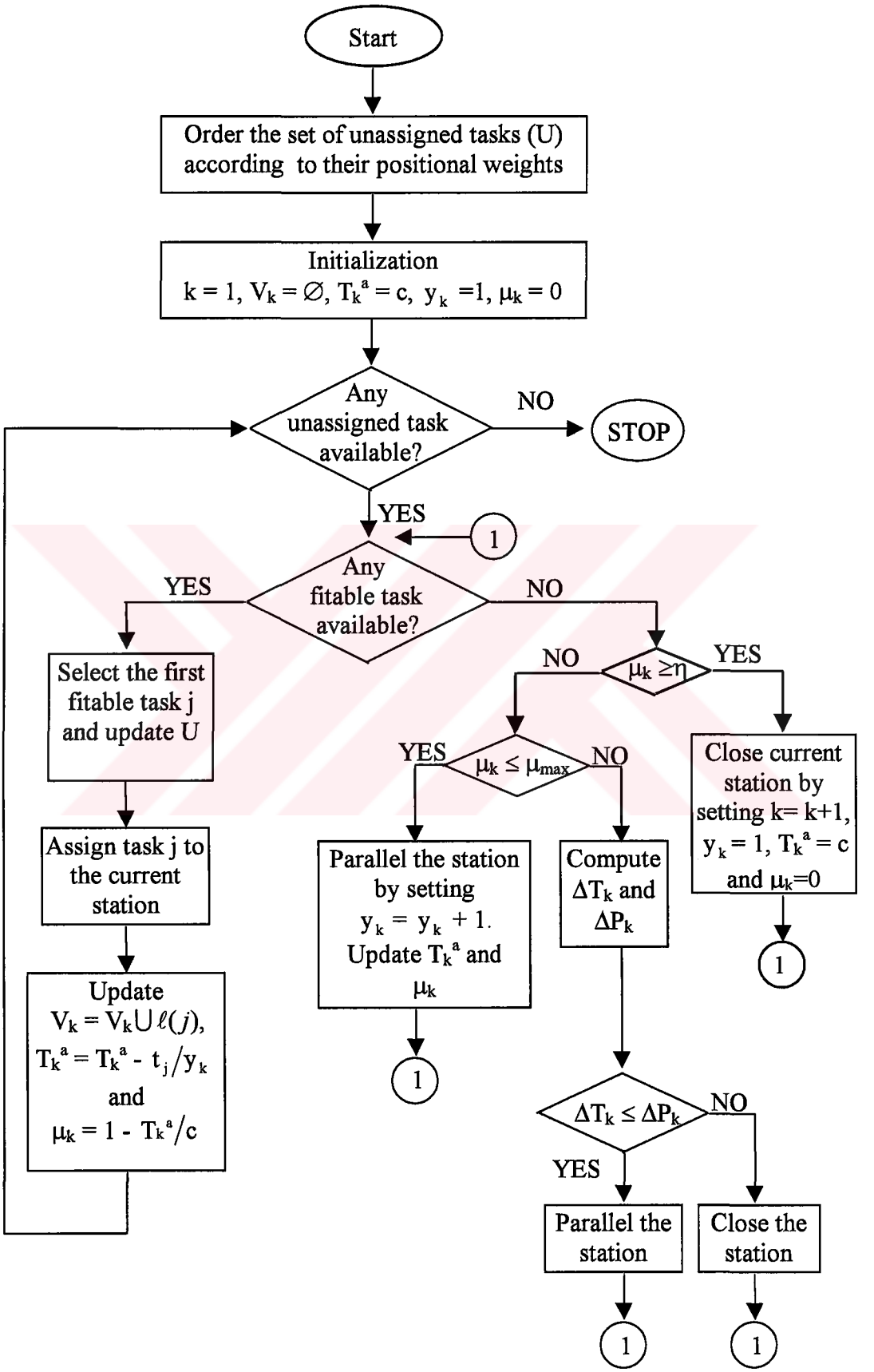


Figure 4.1 Flowchart of Askin and Zhou' s Heuristic Algorithm

4.2.2 Lower Bounds

In this section we present a lower bound on total cost that is calculated for each node of the branch and bound tree. Our lower bound, LB_C, consists of two componenets: a lower bound on the number of workstations and a lower bound on the equipment cost. We calculate LB_C as follows:

$LB_C = L^*LB_S + LB_T$ where,

LB_S: Lower bound on the number of stations

LB_T: Lower bound on equipment cost

A lower bound on number of stations for the unassigned tasks can be calculated by dividing the sum of the processing times of these tasks by the cycle time. A lower bound for a particular current node is the sum of the number of stations formed by assigned tasks and the bound of the number of stations for the remaining tasks. However, this is not the case when a station is paralleled. When task j is assigned to stage k by adding parallel stations, the total number of parallel stations at this stage is $\left\lceil \frac{Fultim_k + t_j}{c} \right\rceil$. So, the number of parallel stations added to this stage is equal to $\left\lceil \frac{Fultim_k + t_j}{c} \right\rceil - y_k$. When station paralleling is considered, we add the number of parallel stations added to the total number of stations already formed to find a lower bound. So, LB_S can be represented by two components as LB_s^1 and LB_s^2 where;

LB_s^1 = total number of stations already formed + lower bound on the number of stations for the remaining tasks

LB_s^2 = total number of stations already formed + number of parallel stations added

$$LB_S = \begin{cases} LB_S^1, \text{ if the current stage is closed} \\ LB_S^2, \text{ if parallel stations are added to current stage} \end{cases}$$

A lower bound on tooling cost is calculated by the sum of equipment costs incurred until this node and total equipment cost of unassigned tools. It is important to note that when a station is paralleled the equipment in that station is also duplicated, the additional equipment cost due to duplication is added to the bound. LB_T can be represented by three components LB_T^1, LB_T^2 and LB_T^3 where;

LB_T^1 = total equipment cost incurred until this node + $\left\lceil \frac{t_j}{c} \right\rceil$ *total equipment cost required to perform unassigned tasks

LB_T^2 = total equipment cost incurred until this node + additional equipment cost because of duplication + $\left\lceil \frac{Fultim_k + t_j}{c} \right\rceil$ *total equipment cost of unassigned equipment

LB_T^3 = total equipment cost incurred until this node + y_k * total equipment cost of unassigned equipment

$$LB_T = \begin{cases} LB_T^1, \text{if the current stage is closed} \\ LB_T^2, \text{if parallel stations are added to the current stage} \\ LB_T^3, \text{otherwise} \end{cases}$$

There are three types of lower bound calculations with respect to the type of node additions. These are as follows:

Type I additions: In this case, lower bound on station number equals to the lower bound on station number of its source node at the previous level, i.e. the node from which the selected node is originating. So, there is no need to make any calculation for this bound. Lower bound on tooling cost equals to LB_T^3. Lower bound on total cost is found by using these lower bound values.

Type II additions: In this case, lower bound on station number equals to LB_S^2 and lower bound on total cost is found by L* LB_S^2 + LB_T^2.

Type III additions: In this case, lower bound on station number equals to LB_S^1 and lower bound on tooling cost equals to LB_T^1. Lower bound on total cost is:

$LB_C = L^* LB_S^1 + LB_T^1$

4.2.3 Steps of The Proposed Algorithm

Our branch and bound procedure consists of mainly nine steps. The additional notation used and the step by step definition of the procedure are as follows:

r : Level of the branch and bound tree

PS : Partial sequence of tasks

x : Node index

$Index_x$: Condition indicator

$$Index_x = \begin{cases} 1, \text{if Type I addition} \\ 2, \text{if Type II addition} \\ 3, \text{if Type III addition} \end{cases}$$

$Stime_k$: Station idle time for each station at stage k

EC_k : Realized equipment cost at stage k

TC : Total realized cost

$Fultim_k$: Total time of tasks assigned to each station at stage k

y'_k : Total number of stations at stage k after task j is assigned to this stage which is calculated as $\left\lceil \frac{Fultim_k + t_j}{c} \right\rceil$

DE_x : Total equipment cost required for assigning task j at node x to stage k

Step 1. Initialization

Read the data. Set:

$r=1$, $k=1$, $y_k=1$, $V_k=\varnothing$, $PS=\varnothing$, $TC=0$, $EC_k=0$, $Fultim_k=0$

Step 2. Determine the tasks with no predecessors. For each node x, determine the type of addition, and set the appropriate $Index_x$ value. Note that if r=1, then $Index_x$=3.

Step 3. For each node x calculate AC_x values as follows:

If $Index_x$ is 1, then:

$$DE_x = \begin{cases} A_\ell * y_k, & \text{if the equipment } \ell \text{ required by task j is not available at stage k} \\ 0, & \text{otherwise} \end{cases}$$

$$AC_x = TC + DE_x$$

If $Index_x$ is 2, then:

$$DE_x = \begin{cases} A_\ell * y'_k, & \text{if the equipment } \ell \text{ required by task j is not available at stage k} \\ 0, & \text{otherwise} \end{cases}$$

$$AC_x = TC + L*\left(y'_k - y_k\right) + \frac{EC_k}{y_k} * \left(y'_k - y_k\right) + DE_x$$

If $Index_x$ is 3, then:

$$DE_x = A_\ell * \left\lceil \frac{t_j}{c} \right\rceil$$

$$AC_x = TC + L * \left\lceil \frac{t_j}{c} \right\rceil + DE_x$$

Step 4. Sort the nodes at level r in nondecreasing order of AC_x values. For each node x calculate lower bound on total cost (LB_C) as explained in section 4.2.2. If $LB_C \geq UB_C$ where UB_C is the best known upper bound on total cost for any node, fathom this node.

Step 5. If all nodes are fathomed, go to step 8. Otherwise, select task j at node x with minimum AC_x and add the task to the partial solution.

Step 6. Add the selected task j to the partial solution as follows:

If $Index_x$ is 1, then:

$$Stime_k = Stime_k - t_j / y_k,\ V_k = V_k \cup \ell(j),\ PS = PS \cup (j),$$
$$Fultim_k = Fultim_k + t_j$$

$$TC=\begin{cases} TC + A_\ell * y_k & \text{, if equipment } \ell \text{ required by task j is not available at stage k} \\ TC & \text{, otherwise} \end{cases}$$

$$EC_k=\begin{cases} EC_k + A_\ell * y_k & \text{, if equipment } \ell \text{ required by task j is not available at stage k} \\ EC_k & \text{, otherwise} \end{cases}$$

r=r+1. If r is equal to N+1, update UB_C and go to step 8. Otherwise, go to step 7.

If $Index_x$ is 2, then:

$V_k = V_k \cup \ell(j)$, $PS = PS \cup (j)$

$$TC=\begin{cases} TC+L*(y_k^{'}-y_k)+\dfrac{EC_k}{y_k}*(y_k^{'}-y_k)+A_\ell*y_k^{'} & \text{, if equipment } \ell \text{ required by task j is not available at stage k} \\ TC+L*(y_k^{'}-y_k)+\dfrac{EC_k}{y_k}*(y_k^{'}-y_k) & \text{, otherwise} \end{cases}$$

$$EC_k=\begin{cases} \dfrac{EC_k}{y_k}*y_k^{'}+A_\ell*y_k^{'} & \text{, if equipment } \ell \text{ required by task j is not available at stage k} \\ \dfrac{EC_k}{y_k}*y_k^{'} & \text{, otherwise} \end{cases}$$

$$Fultim_k = Fultim_k + t_j,\ y_k = \left\lceil \frac{Fultim_k}{c} \right\rceil,$$

$Stime_k = c - Fultim_k / y_k$, r = r + 1. If r is equal to N+1, update UB_C and go to step 8. Otherwise, go to step 7.

If $Index_x$ is 3, then:

$$k=\begin{cases} k & \text{, if r is equal to 1} \\ k+1 & \text{, otherwise} \end{cases}$$

$y_k=\left\lceil \frac{t_j}{c} \right\rceil$, $Stime_k$=c, V_k=∅, EC_k=0, $Fultim_k$=0

$Stime_k = Stime_k - t_j / y_k$, $V_k = V_k \cup \ell(j)$, $PS = PS \cup (j)$

$EC_k = EC_k + A\ell * y_k$, $Fultim_k = Fultim_k + t_j$, $TC = TC + L * y_k + EC_k$, r=r+1. If r is equal to N+1, update UB_C and go to step 8. Otherwise, go to step 7.

Step 7. Determine fitable tasks to the current stage. Set $Index_x$ value as 1 for each node and go to step 3. If there is not any fitable task, then consider Type II and Type III additions simultaneously and determine fitable tasks for both of these type of additions. Set appropriate $Index_x$ value for each node and go to step 3.

Step 8. Backtrack to the previous level.

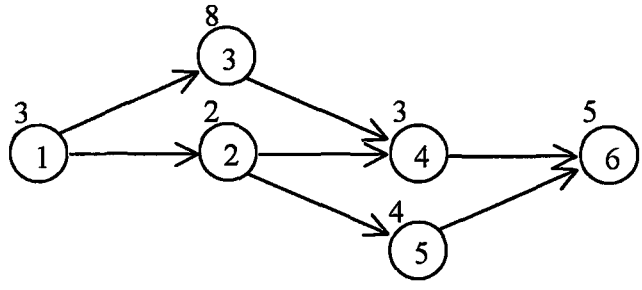
i. r=r-1. If r is equal to zero, go to step 9.

ii. Remove the task at the current node from the partial solution. Update PS, k, y_k, V_k, $Stime_k$, $Fultim_k$, EC_k and TC. If there is an unsearched node at this level, select the task at this node to add to the partial solution and go to step 6.

Otherwise, go to step i.

Step 9. Stop.

We illustrate our branch and bound algorithm through Example 3. The precedence diagram of the assembly line is given in Figure 4.2. The cycle time is 6 minutes.



Assume L=50 and average equipment cost, $A\ell$, is 5. The equipment type required by each task is as follows:

Task (j)	Type of equipment (ℓ)
1	1
2	2
3	3
4	4
5	1
6	5

Assume we have not got an initial solution. So, the first complete branch of the tree provides an upper bound for the problem. The first branch of the tree is as follows:

The only task with no predecessor is task 1. So, we add task 1 to the partial solution. Since it is Type III addition, we start with 1 station at stage 1. Accumulated cost and lower bound calculations are as follows:

$$DE_1 = A_\ell * \left\lceil \frac{t_1}{c} \right\rceil \Rightarrow DE_1 = 5*1 \Rightarrow DE_1 = 5$$

$$AC_1 = L * \left\lceil \frac{t_1}{c} \right\rceil + DE_1 \Rightarrow AC_1 = 50*1+5*1 \Rightarrow AC_1 = 55$$

$$LB_S = \left\lceil \frac{\sum_{j=1}^{6} t_j}{c} \right\rceil \Rightarrow LB_S = 5$$

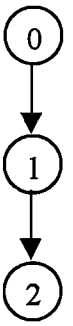
$$LB_T = \left\lceil \frac{t_1}{c} \right\rceil * \sum_{\ell=1}^{5} A_\ell \Rightarrow LB_T = 25$$

$$LB_C = L * LB_S + LB_T \Rightarrow LB_C = 50*5+25 \Rightarrow LB_C = 275$$

We assign task 1 to the first stage with its equipment. The related calculations are as follows:

$k=1$, $y_1=1$, $V_1=\{1\}$, $PS=\{1\}$, $Stime_1=3$, $Fultim_1=3$, $EC_1=5$, $TC=55$

The second level of the branch and bound tree is as follows:



The only fitable task to stage 1 is task 2. So, we add task 2 to the partial solution. Since it is a Type I addition, accumulated cost and lower bound calculations are as follows:

$$DE_2 = A_\ell * y_1 \Rightarrow DE_2 = 5$$

$$AC_2 = TC + DE_2 \Rightarrow AC_2 = 55 + 5 \Rightarrow AC_2 = 60$$

Because it is a Type 1 addition, LB_S for this node is equal to the lower bound on station number of its source node, i.e.node 1. So, LB_S for this node is 5.

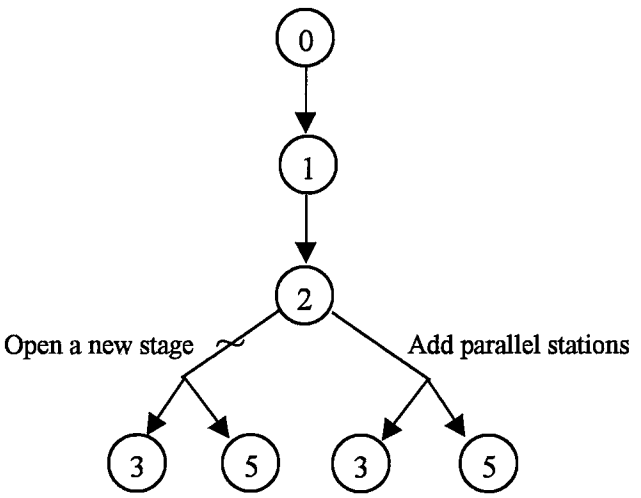
$$LB_T = 5 + \sum_{\ell=2}^{5} A_\ell \Rightarrow LB_T = 25$$

$$LB_C = L * LB_S + LB_T \Rightarrow LB_C = 275$$

We assign task 2 to the first stage with its equipment. The related calculations are as follows:

$Stime_1=1$, $V_1=\{1,2\}$, $PS=\{1,2\}$, $Fultim_1=5$, $TC=60$, $EC_1=10$

The third level of the branch and bound tree is as follows:



Because there is no fitable task for stage 1, we both consider opening a new stage or adding parallel stations to stage 1 simultaneously. The tasks with no unassigned predecessors are 3 and 5. Node 3 and node 4 represent tasks 3 and 5, respectively. Both of them are Type III additions. Because the processing time of task 3 is greater than the cycle time, a new stage is begun with 2 stations. Accumulated cost and lower bound calculations for node 3 are as follows:

$$DE_3 = A_\ell * \left\lceil \frac{t_3}{c} \right\rceil \Rightarrow DE_3 = 5*2 \Rightarrow DE_3 = 10$$

$$AC_3 = TC + L * \left\lceil \frac{t_3}{c} \right\rceil + DE_3 \Rightarrow AC_3 = 60 + 50*2 + 10 \Rightarrow AC_3 = 170$$

$$LB_S = 1 + \left\lceil \frac{\sum_{j=3}^{6} t_j}{c} \right\rceil \Rightarrow LB_S = 5$$

$$LB_T = 10 + \left\lceil \frac{t_3}{c} \right\rceil * 20 \Rightarrow LB_T = 50$$

$$LB_C = 50*5 + 50 \Rightarrow LB_C = 300$$

Accumulated cost and lower bound calculations for node 4 are as follows:

$$DE_4 = A_\ell * \left\lceil \frac{t_5}{c} \right\rceil \Rightarrow DE_4 = 5$$

$$AC_4 = 60 + 50 * 1 + 5 = 115$$

$$LB_S = 1 + \left\lceil \frac{\sum_{j=3}^{6} t_j}{c} \right\rceil \Rightarrow LB_S = 5$$

$$LB_T = 10 + 1 * 20 \Rightarrow LB_T = 30$$

$$LB_C = 50 * 5 + 30 \Rightarrow LB_C = 280$$

Nodes 5 and 6 represent the assignment of tasks 3 and 5, respectively. But this time, both of them are Type II additions. When task 3 is assigned to stage 1, 2 parallel stations are added to the stage. Accumulated cost and lower bound calculations for node 5 are as follows:

$$DE_5 = 5 * \left\lceil \frac{Fultim_k + t_3}{c} \right\rceil = 15$$

$$AC_5 = 60 + 50 * 2 + 10 * 2 + 15 \Rightarrow AC_5 = 195$$

$$LB_S = 1 + 2 \Rightarrow LB_S = 3$$

$$LB_T = 10 + 20 + 45 \Rightarrow LB_T = 75$$

$$LB_C = 50 * 3 + 75 = 225$$

When task 5 is assigned to stage 1, 1 parallel station is added to this stage. Accumulated cost and lower bound calculations for node 6 are as follows:

$DE_6 = 0$, since the equipment required by task 5 is available at stage 1.

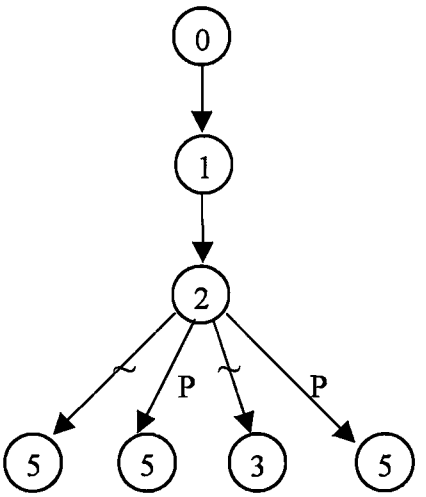
$$AC_6 = 60 + 50 + 10 \Rightarrow AC_6 = 120$$

$$LB_S = 1 + 1 \Rightarrow LB_S = 2$$

$$LB_T = 10 + 10 + 30 \Rightarrow LB_T = 50$$

$$LB_C = 50 * 2 + 50 = 150$$

If the nodes are sorted in increasing order of AC_x , the branch and bound tree becomes as follows:



So, the decision given in third level is that begin a new stage and assign task 5 to this stage.

The nodes are generated through the algorithm in the same manner. After searching all nodes, the optimal solution is reached.

CHAPTER 5

COMPUTATIONAL RESULTS

We have experimented with our branch and bound algorithm to investigate the effects of changing certain parameter values on some performance measures. We have also compared our algorithm with the heuristic of Askin and Zhou (1997) in terms of total cost. In this chapter, we first define our performance measures, followed in section 5.2 by a description of experimental parameters and generation of test problems. Pilot run results are given in section 5.3, and production run results are discussed in section 5.4.

5.1 PERFORMANCE MEASURES

In evaluating our branch and bound algorithm, we use five performance measures. Both average and maximum (worst case) values are calculated for each measure. These performance measures are as follows:

1. CPU Time: This is a measure of running (or elapsed) time of the algorithm. CPU time is given in minutes.

2. Total Number of Nodes Generated: This measures the size of the branch and bound tree.

3. Total Number of Nodes Generated Until Reaching The Optimal Solution: This measure tells us how soon the optimal solution is reached and gives an idea about the efficiency of lower bound values.

4. Total Cost: This is the optimal total cost required to balance the line including both total station opening cost and total equipment cost.

5. Percentage Deviation of the Optimal Solution from Heuristic Solution in Terms of Total Cost: This measure will be referred to as "Percentage Deviation" thereafter. It enables us to compare Askin and Zhou' s heuristic method with our algorithm. Percentage deviation is calculated as follows:

$$\text{Percentage Deviation} = \frac{\text{Total Cost (found by heuristic)} - \text{Optimal Total Cost}}{\text{Optimal Total Cost}}$$

5.2 EXPERIMENTAL PARAMETERS AND PROBLEM GENERATION

In order to measure the efficiency of our algorithms, randomly problems are generated and solved. The characteristics used in problem generation are based on an examination of similar studies available in the literature. Most of these characteristics also used to set the experimental parameters we have used in this study. Details of problem characteristics are as follows:

1. *Problem Size:* Assembly networks are randomly generated for different numbers of tasks. Number of tasks is set to 10, 15, 20, 25, 30, 35, 40, 50, 60, and to 70 in one case.

2. *Distribution of task times:* In most studies, uniform distribution is used for generating task times. Hence, we set the task times to uniformly distributed values between 10 and 100.

3. *Cycle time:* Three values are considered for cycle time. These are maxPT, 1.4*maxPT and 1.8*maxPT where maxPT represents maximum task time in an assembly network.

4. *Flexibility Ratio (FR):* This ratio affects the number of alternative production lines that may be established with the same set of tasks. The values of FR close to 0 indicate a highly interconnected network, and the values of FR close to 1

indicate a more flexible network with fewer precedence relations. In our study, FR values of 0.2, 0.5 and 0.8 are used for problem generation.

5. *Cost structure:* The changes in cost structure can be reflected by the ratio of L/A_ℓ. This ratio is set to 30/30 and 30/1. When L/A_ℓ equals to 30/30, this means that equipment cost is comparable to station opening cost. With this setting, the problem becomes a simultaneous optimization problem, i.e. both station opening and equipment costs must be optimized simultaneously. When L/A_ℓ is 30/1, equipment cost is much lower relative to station opening cost. Hence, hierarchical optimization is aimed with this setting, i.e. first station opening cost is optimized, followed by equipment cost. Station opening cost is fixed at \$30000. Average equipment cost for all types of equipment is either \$30000 or \$1000 according to L/A_ℓ ratio.

6. *Task equipment types:* The total number equipment types are generated from discrete uniform distribution between 1 and the number of tasks. If the total number of equipment types generated for a particular assembly network is represented by D as in the mathematical formulation, then the type of equipment required by any task in the network is distributed as discrete uniform between 1 and D.

Several methods for generating a precedence-ordering relation on a set of tasks can be devised, each resulting in the same problem density. It will be useful to explain the partial-order generating method that we have used to generate precedence-ordering relations.

As explained before, an assembly network can be represented by a precedence matrix that indicates the precedence relations. Each cell above the diagonal has a value of 0 or 1. If task i precedes task j, then the value of respective cell is 1. Otherwise, it is 0. Our method uses this precedence matrix to generate networks with desired F-ratio. First of all, the number of zeros in the precedence matrix is calculated by the following expression where N is the number of tasks:

$$\text{number of zeros} = \frac{\text{Desired FR*N*(N-1)}}{2}$$

As we know, there are N*(N-1)/2 cells above the diagonal. Let us show this quantity by NC. After finding the total number of zeros, the value of NC is calculated, and all the cells above the diagonal beginning with the first entry are numbered between 1 and NC sequentially. A simple example of a precedence matrix with 4 tasks is shown below after numbering of the cells.

	1	2	3	4
1	-	1	2	3
2		-	4	5
3			-	6
4				-

Then, a random number is generated from discrete uniform distribution between 1 and NC, and a 0 is placed in the cell having the same cell number as this. After this first entry, since there are NC-1 cells with no entries, these cells are renumbered between 1 and NC-1 in the same manner. For example, if the random number generated between 1 and 6 is 4 for the above precedence matrix, the new form of the precedence matrix will be as follows:

	1	2	3	4
1	-	1	2	3
2		-	0	4
3			-	5
4				-

This time, a random number is generated between 1 and NC-1 and a 0 is placed into the cell having the same value as this random variate. This random variate generation and renumbering procedure continues until the required number of

zeros in the precedence matrix is reached. Then, 1 is entered into all remaining cells above the diagonal, and thus a random network is constructed.

Among the networks we have generated randomly, most have the same FR as the desired FR. Only a few of them show little differences. Those are the ones with problem sizes equal to 10, 15, 30, 35 and 50 when FR ratios for those cases are found to be 0.49.

Main experimental parameters used in this study are the number of tasks, cycle time, FR and cost structure. The effects of changing these parameters on the performance measures are investigated. Once a random problem is generated, values of the remaining parameters are fixed, and kept at the same value during the runs. These parameters are task times, required type of equipment for each task, and precedence relations.

Ten problems are randomly generated for each combination of L/A_ℓ, FR and number of tasks. The same problem is solved with each cycle time level. Each problem is firstly solved by the modified heuristic of Askin and Zhou (1997) with μ_{max} equals to 0.5 and η equals to 0.85 and then by our algorithm. In both algorithms, maximum three-level station paralleling is allowed for each problem as will be explained in section 5.3. If optimal solution cannot be found by the branch and bound algorithm within 180 minutes time limit, the program is terminated with the best solution found thus far. The problems are solved on computers with 550 MHz Pentium III processor and 64-MB memory. All programs are coded in Microsoft Fortran Powerstation version.

5.3 PILOT RUNS

A number of problems whose optimal solutions are known are taken from the open literature. At the internet address *http:\\www.bwl.tu-darmstadt.de/bwl3/forsch/projekte/alb,* Armin Scholl and Robert Klein present benchmark data

sets for SALB. Hence, we have used the problems taken from this source to test our program.

The problems solved have different characteristics. The number of tasks varies between 7 and 53. Each problem set is named with the name of the author who used it. The characteristics of each problem are given in Table 5.1.

Table 5.1 Characteristics of the test problems

Name	# of tasks	Min. task time	Max. task time	Order Strength
Bowman	8	3	17	75.00
Buxey	29	1	25	50.74
Gunther	35	1	40	59.50
Hahn	53	40	1775	83.82
Heskiaoff	28	1	108	22.49
Jackson	11	1	7	58.18
Jaeschke	9	1	6	83.33
Kilbridge	45	3	55	44.55
Lutz1	32	100	1400	83.47
Mansoor	11	2	45	60.00
Mertens	7	1	6	52.38
Mitchell	21	1	13	70.95
Roszieg	25	1	13	71.67
Sawyer	30	1	25	44.83

As these problems are SALB, paralleling of stations is not allowed. Hence, we have not allowed station paralleling when solving these problems. The optimal solution of each problem represents the minimum number of workstations required to balance the line. No equipment is considered in any of the original problems. We assume 3 types of equipment for each problem. The equipment is assigned to tasks in such a manner that the task with number 1 requires tooling type 1, the second task requires tooling type 2, the third task requires tooling type 3, and after that the equipment type cycles. Station opening cost is set to 100 and average equipment cost is set to 10.

Optimal solutions of the test problems with different cycle time values are given in Appendix A. We have found the same optimal solutions with our branch and bound algorithm.

We have also made some pilot runs to decide on the maximum level of paralleling. For number of tasks 10 and 15, randomly generated ten problems were solved with maximum level of paralleling 2, 3 and 4. After these runs it has been seen that, although total number of nodes generated increases drastically, four parallel stations are opened very rarely. Therefore, we have fixed maximum level of paralleling as three.

5.4 DISCUSSION OF RESULTS

5.4.1 Effects of Experimental Parameters

Our results concerning the effects of experimental parameters are summarized in Tables 5.2-5.7 for different combinations of L/A_ℓ and FR. In the tables, average and maximum values over ten problems are given for our performance measures. Opt.Nodes/Total Nodes gives the ratio of number of nodes generated in branch and bound algorithm until optimal solution is found to the total number of nodes generated until the algorithm stops.

In Table 5.2, when L/A_ℓ is 30/30 and FR is 0.2, average and maximum values of performance measures are given for different problem sizes (N) and cycle times (CT). Increasing CT decreases the average and maximum values of number of nodes, CPU time and total cost. When CT is increased, more tasks can be assigned to a station, and a balance can be achieved with fewer stations and therefore with lower total cost. Since increasing the cycle time yields more efficient lower bound values, more nodes are fathomed. Consequently, fewer nodes are generated and less CPU time is required to reach the optimal solution.

Table 5.2 Results for L/ A_ℓ =30/30, FR=0.2

		CPU Time (min.)		Total # of Nodes		# of Nodes Until Opt.		Opt. Nodes/Total Nodes		Total Cost(*1000)	
N	CT(*maxPT)	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	0.0000	0.0000	119.5	261.0	33.5	135.0	0.28	0.80	480.0	600.0
	1.4	0.0000	0.0000	49.9	105.0	11.2	58.0	0.19	1.00	396.0	540.0
	1.8	0.0000	0.0000	35.2	71.0	7.7	24.0	0.23	0.79	327.0	420.0
15	1	0.0000	0.0000	2,393.8	13,203.0	1,230.0	11,465.0	0.28	0.89	732.0	900.0
	1.4	0.0000	0.0000	582.1	3,501.0	191.5	1,065.0	0.28	0.91	624.0	720.0
	1.8	0.0000	0.0000	177.1	772.0	34.9	145.0	0.30	0.93	525.0	600.0
20	1	0.0017	0.0167	17,879.3	91,643.0	1,124.1	10,826.0	0.11	0.32	1,026.0	1,170.0
	1.4	0.0000	0.0000	2,020.1	8,428.0	490.2	2,792.0	0.19	0.43	837.0	960.0
	1.8	0.0000	0.0000	643.9	2,696.0	54.8	278.0	0.06	0.31	711.0	870.0
25	1	0.0050	0.0167	16,009.6	120,423.0	899.6	8,259.0	0.16	0.71	1,278.0	1,410.0
	1.4	0.0000	0.0000	2,367.6	14,510.0	1,271.8	9,923.0	0.33	0.90	1,113.0	1,230.0
	1.8	0.0000	0.0000	588.5	2,965.0	325.3	2,075.0	0.31	0.83	966.0	1,050.0
30	1	0.0233	0.0833	237,466.5	803,763.0	20,907.2	197,256.0	0.05	0.38	1,530.0	1,620.0
	1.4	0.0017	0.0167	23,747.7	99,723.0	5,384.4	26,410.0	0.27	0.90	1,317.0	1,470.0
	1.8	0.0000	0.0000	5,654.0	13,728.0	759.6	7,367.0	0.07	0.55	1,140.0	1,320.0
35	1	0.0100	0.0333	102,979.5	502,816.0	681.6	3,966.0	0.03	0.16	1,824.0	1,950.0
	1.4	0.0017	0.0167	14,598.6	67,416.0	85.5	232.0	0.05	0.22	1,617.0	1,770.0
	1.8	0.0000	0.0000	2,102.4	11,120.0	126.5	694.0	0.12	0.35	1,401.0	1,470.0
40	1	1.7000	7.7800	15,911,350.0	75,142,077.0	3,330,690.1	28,450,268.0	0.12	0.53	2,096.7	2,310.0
	1.4	0.1567	1.0167	1,370,308.0	8,991,960.0	176,034.2	1,493,975.0	0.12	0.72	1,734.0	1,920.0
	1.8	0.0217	0.1000	181,044.6	799,418.0	51,407.8	511,974.0	0.07	0.71	1,509.0	1,650.0
50	1	2.7800	12.1833	15,070,097.9	62,318,113.0	9,850.6	86,074.0	0.01	0.04	2,604.0	2,700.0
	1.4	0.1433	0.9667	950,008.6	6,165,460.0	90,726.8	688,587.0	0.09	0.61	2,280.0	2,370.0
	1.8	0.0167	0.1167	100,169.2	706,639.0	38,080.6	370,334.0	0.09	0.52	1,998.0	2,040.0
60	1	10.2300	65.9167	61,870,658.3	404,171,440.0	7,452,502.2	63,946,583.0	0.10	0.58	3,120.0	3,180.0
	1.4	0.0950	0.2833	564,677.1	1,703,710.0	28,736.2	196,887.0	0.04	0.22	2,745.0	2,850.0
	1.8	0.0167	0.0667	93,293.6	401,646.0	1,271.5	12,424.0	0.01	0.07	2,451.0	2,520.0
70	1	94.9750	180.0667	538,784,455.9	1,197,000,000.0	26,556,303.0	257,907,813.0	0.03	0.29	3,615.0	3,780.0
	1.4	7.7722	62.0500	38,893,651.6	306,811,620.0	260,417.4	1,520,247.0	0.02	0.11	3,186.7	3,330.0
	1.8	0.4093	3.0167	2,018,599.8	14,426,393.0	111,351.7	892,815.0	0.13	0.48	2,830.0	2,910.0
40	1	13.5283	119.9667	115,395,538.0	1,010,753,229.0	16,299,617.2	133,019,961.0	0.12	0.53	2,043.0	2,310.0
70	1.4	25.0000	180.0500	133,754,286.4	987,500,000.0	29,543,119.8	293,087,441.0	0.05	0.30	3,135.0	3,330.0
	1.8	3.5833	32.1500	16,458,436.2	146,416,964.0	920,303.9	8,200,874.0	0.12	0.48	2,754.0	2,910.0

When cycle time is kept constant, increasing the number of tasks yields an increase in the total cost. It is expected that, when N is increased for a fixed value of CT, the number of nodes increases and more CPU time is spent to reach the optimal solution. However, it is interesting that an increase in N does not always yield an increase in the number of nodes and CPU time as seen in Table 5.2. Only when the difference between two N values is high, the increase in number of nodes and CPU time becomes obvious. The same is not true for close N values. For example when CT is maxPT, increasing N from 30 to 35 results in a decrease in the average number of nodes and in the average CPU time. This can only be explained as a random effect.

As seen in Table 5.2, an extreme case is observed when N is 40 and CT is maxPT. Among ten problems that are solved at this combination, problem 8 shows a very large difference from the others in terms of the CPU time and the number of nodes. When N is 70, we encounter with another extreme case. Problem 6 gives very different solutions when CT is 1.4*maxPT and 1.8*maxPT, yielding uncharacteristically large average and maximum values. Treating these as outliers, the results of these problems are disregarded when calculating the average and maximum values of performance measures in the upper rows. The original average and maximum values of performance measures including the outlier problems are given at the bottom of the table with highlighted N values of 40 and 70.

Table 5.3 shows the average and maximum values of performance measures for different values of N and CT when L/A_ℓ is 30/30 and FR is 0.5. The effects of increasing CT and N are similar to those observed in Table 5.2. When Table 5.2 and Table 5.3 are compared for the same values of L/A_ℓ, CT and N, as FR is increased from 0.2 to 0.5, the number of nodes and the CPU time increase, but total cost decreases. When FR is increased, i.e. when there are fewer precedence relations, the assembly line becomes more flexible and the number of candidate tasks for a station increases. This causes generation of more nodes, and more CPU time is required to reach the optimal solution. As a result, we were not able to

Table 5.3 Results for L/ A_ℓ =30/30, FR=0.5

		CPU Time (min.)		Total # of Nodes		# of Nodes Until Opt.		Opt. Nodes/Total Nodes		Total Cost(*1000)	
N	CT(*maxPT)	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	0.0000	0.0000	672.4	2,144.0	81.0	294.0	0.18	0.88	465.0	570.0
	1.4	0.0000	0.0000	352.6	1,132.0	96.1	693.0	0.22	0.68	378.0	480.0
	1.8	0.0000	0.0000	231.9	801.0	25.8	165.0	0.13	0.68	318.0	420.0
15	1	0.0000	0.0000	6,360.5	22,523.0	1,591.0	6,030.0	0.38	0.83	705.0	840.0
	1.4	0.0000	0.0000	2,464.0	8,675.0	642.5	5,617.0	0.22	0.65	603.0	720.0
	1.8	0.0000	0.0000	665.0	2,099.0	30.7	60.0	0.12	0.58	501.0	630.0
20	1	0.0317	0.2167	462,053.2	3,056,359.0	113,074.2	1,126,079.0	0.14	0.56	960.0	1,140.0
	1.4	0.0033	0.0167	40,167.2	177,941.0	16,057.4	145,666.0	0.31	0.82	807.0	900.0
	1.8	0.0000	0.0000	7,140.4	36,086.0	927.1	6,785.0	0.35	0.73	660.0	840.0
25	1	0.1650	1.3167	2,808,505.9	22,747,501.0	1,999,215.7	19,648,763.0	0.18	0.86	1,242.0	1,320.0
	1.4	0.0017	0.0167	43,852.8	168,634.0	19,525.8	72,114.0	0.38	0.78	1,062.0	1,170.0
	1.8	0.0000	0.0000	10,124.3	55,603.0	5,005.3	45,900.0	0.20	0.83	921.0	1,020.0
30	1	2.7367	11.0833	35,823,782.0	143,263,860.0	1,241,188.9	6,284,101.0	0.12	0.84	1,482.0	1,620.0
	1.4	0.1850	0.7500	2,251,256.2	8,291,700.0	219,686.4	1,523,876.0	0.17	0.63	1,248.0	1,380.0
	1.8	0.0617	0.5000	885,443.6	7,567,334.0	35,429.1	285,484.0	0.10	0.33	1,044.0	1,140.0
35	1	9.1600	59.1667	129,379,206.1	922,961,684.0	7,423,981.2	72,858,697.0	0.13	0.65	1,794.0	1,950.0
	1.4	0.1600	0.5333	1,997,559.2	6,608,997.0	328,156.3	2,676,531.0	0.12	0.53	1,533.0	1,620.0
	1.8	0.0233	0.1000	308,672.8	1,018,375.0	93,831.8	753,660.0	0.26	0.86	1,323.0	1,410.0
40	1	66.7433	180.0167	707,766,239.0	1,989,500,000.0	20,132,313.0	178,857,484.0	0.04	0.29	1,989.0	2,250.0
	1.4	20.2433	180.0167	183,535,973.7	1,546,000,000.0	6,934,260.6	52,208,246.0	0.17	0.65	1,665.0	1,890.0
	1.8	5.7350	55.1833	49,681,786.4	472,041,765.0	421,652.6	2,618,814.0	0.14	0.38	1,428.0	1,590.0
50	1	112.8267	180.6500	1,106,571,378.9	2,120,000,002.0	31,245,999.8	310,188,735.0	0.02	0.18	2,556.0	2,640.0
	1.4	41.9633	180.0000	441,456,985.5	1,899,500,005.0	12,326,182.6	52,906,509.0	0.05	0.47	2,184.0	2,250.0
	1.8	3.7750	28.9333	36,660,172.4	286,547,746.0	1,557,661.7	14,807,671.0	0.01	0.05	1,911.0	2,040.0

solve problems of size 60 and 70 when FR is 0.5. However, since the assembly line is more flexible, a better balance with less total cost is obtained.

Table 5.4 shows the values of performance measures for different values of N and CT when L/A_ℓ is 30/30 and FR is 0.8. Again, increasing CT decreases the number of nodes, CPU time and total cost. When N equals to 20, we encounter with another extreme case. Problem 9 gives very different solutions from the others for each value of CT in terms of the CPU time and the number of nodes. Therefore, we disregard problem 9 when calculating the average and maximum values of performance measures in the upper row. Original values of performance measures including the results of problem 9 are shown at the bottom of the table.

When FR is 0.8, increasing the number of tasks causes a sharper increase in the number of nodes generated and the CPU time. This is an important result for us since, with FRs 0.2 and 0.5, we have not seen such sharp increases in the number of nodes as N increases. Hence, we can conclude that for highly flexible lines, when equipment cost is comparable to station opening cost, increasing the number of tasks makes the problem more difficult. As a result, we were able to solve problems of size up to 30 when FR is 0.8.

The results obtained when L/A_ℓ is 30/1 are summarized in Tables 5.5, 5.6 and 5.7 for FR values of 0.2, 0.5 and 0.8. We have another extreme case (problem 9) in Table 5.7. The effects of increasing CT, increasing N, and changing FR are similar to those observed with L/A_ℓ is 30/30. Comparing Tables 5.2-5.4 and 5.5-5.7 respectively, we see that there is a significant reduction in the total cost when L/A_ℓ is 30/1 because the equipment cost is decreased from 30000 to 1000.

The results show that the most important parameter affecting the values of performance measures is the FR. When the other parameters are kept constant and only the FR is increased, the number of nodes and CPU time increase exponentially. This exponential increase is more obvious for N≥20. Since a

Table 5.4 Results for L/ A_ℓ =30/30, FR=0.8

		CPU Time (min.)		Total # of Nodes		# of Nodes Until Opt.		Opt. Nodes/Total Nodes		Total Cost(*1000)	
N	CT(*maxPT)	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	0.0000	0.0000	32,058.5	105,925.0	1,803.6	10,957.0	0.15	0.70	432.0	570.0
	1.4	0.0000	0.0000	9,105.7	56,945.0	403.5	1,788.0	0.18	0.59	348.0	450.0
	1.8	0.0000	0.0000	3,615.3	24,771.0	227.5	837.0	0.29	0.50	288.0	390.0
15	1	0.0233	0.1167	643,690.2	2,703,030.0	37,787.7	241,267.0	0.09	0.28	666.0	840.0
	1.4	0.0067	0.0167	94,598.4	385,439.0	20,727.2	101,134.0	0.16	0.46	540.0	690.0
	1.8	0.0017	0.0167	19,169.4	75,955.0	4,352.2	30,687.0	0.25	0.71	441.0	570.0
20	1	16.8796	75.8700	361,741,654.3	1,635,363,691.0	85,657,108.0	762,623,728.0	0.12	0.50	883.3	1,080.0
	1.4	0.2148	1.0500	5,004,795.0	24,508,616.0	507,400.6	3,423,839.0	0.29	0.96	710.0	810.0
	1.8	0.0296	0.2500	752,211.0	5,906,568.0	177,521.0	1,519,325.0	0.27	0.91	603.3	720.0
25	1	79.4967	180.0000	1,498,752,603.2	3,769,223,755.0	482,248,341.6	3,091,207,205.0	0.15	0.82	1,176.0	1,320.0
	1.4	2.1733	10.6833	45,322,542.0	196,896,764.0	14,704,669.5	118,584,680.0	0.28	0.85	954.0	1,080.0
	1.8	0.4417	3.7333	8,752,233.7	70,575,913.0	325,971.5	2,349,130.0	0.42	0.95	807.0	900.0
30	1	180.0000	180.0000	3,606,192,907.2	4,156,309,895.0	79,259,586.2	484,165,572.0	0.02	0.13	1,374.0	1,560.0
	1.4	90.4767	180.0000	1,808,490,460.7	3,730,261,163.0	776,379,481.9	3,544,937,898.0	0.40	0.97	1,032.0	1,260.0
	1.8	19.6733	113.3500	425,112,182.9	2,116,682,913.0	261,450,301.0	1,876,077,668.0	0.43	0.89	849.0	1,050.0
20	1	33.1917	180.0000	635,952,254.7	3,103,847,658.0	78,434,044.9	762,623,728.0	0.11	0.50	879.0	1,080.0
	1.4	10.1967	100.0333	189,267,493.0	1,847,631,775.0	527,179.9	3,423,839.0	0.26	0.96	699.0	810.0
	1.8	11.4767	114.5000	231,081,764.6	2,304,047,747.0	159,778.8	1,519,325.0	0.24	0.91	591.0	720.0

Table 5.5 Results for L/ A_ℓ =30/1, FR=0.2

		CPU Time (min.)		Total # of Nodes		# of Nodes Until Opt.		Opt. Nodes/Total Nodes		Total Cost(*1000)	
N	CT(*maxPT)	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	0.0000	0.0000	112.1	396.0	20.0	110.0	0.22	0.73	211.9	255.0
	1.4	0.0000	0.0000	52.3	100.0	8.1	31.0	0.25	0.84	153.2	193.0
	1.8	0.0000	0.0000	43.2	71.0	15.9	48.0	0.35	0.92	122.3	159.0
15	1	0.0000	0.0000	865.2	3,623.0	586.3	3,072.0	0.48	0.85	303.9	357.0
	1.4	0.0000	0.0000	554.6	2,096.0	264.5	1,885.0	0.36	0.90	223.9	261.0
	1.8	0.0000	0.0000	255.7	1,043.0	37.9	243.0	0.23	0.87	175.5	203.0
20	1	0.0000	0.0000	7,684.0	50,631.0	1,105.8	3,887.0	0.36	0.79	414.1	484.0
	1.4	0.0000	0.0000	806.0	3,359.0	433.3	3,166.0	0.46	0.94	296.8	351.0
	1.8	0.0000	0.0000	442.5	1,296.0	84.3	399.0	0.27	0.88	229.8	270.0
25	1	0.0033	0.0333	36,476.1	217,448.0	7,480.4	31,890.0	0.39	0.95	509.8	594.0
	1.4	0.0000	0.0000	2,810.7	6,971.0	1,605.4	5,093.0	0.49	1.00	369.2	408.0
	1.8	0.0000	0.0000	1,638.1	4,187.0	903.7	2,206.0	0.50	0.97	292.7	324.0
30	1	0.0567	0.3833	486,015.3	3,492,780.0	69,347.1	341,208.0	0.37	0.83	617.4	700.0
	1.4	0.0017	0.0167	12,323.2	45,906.0	5,632.7	24,207.0	0.35	0.83	441.2	494.0
	1.8	0.0000	0.0000	6,333.5	16,128.0	3,348.8	8,797.0	0.50	0.82	347.7	387.0
35	1	0.1083	0.3333	885,459.7	2,447,144.0	424,171.5	1,762,017.0	0.67	0.98	726.7	830.0
	1.4	0.0050	0.0167	30,729.0	105,021.0	4,429.2	18,773.0	0.19	0.47	525.4	593.0
	1.8	0.0017	0.0167	19,482.3	129,173.0	2,891.8	13,682.0	0.23	0.82	411.1	462.0
40	1	1.6450	5.6167	12,184,000.7	43,738,955.0	1,984,278.2	12,603,553.0	0.45	0.84	819.8	959.0
	1.4	0.0667	0.2333	476,530.7	1,916,168.0	221,574.0	1,222,662.0	0.59	0.97	590.4	683.0
	1.8	0.0300	0.1833	197,284.6	1,179,311.0	82,581.1	727,115.0	0.45	0.89	464.5	529.0
50	1	8.0133	18.8500	46,131,473.2	110,624,158.0	24,547,981.8	71,921,632.0	0.56	0.91	1,017.6	1,115.0
	1.4	0.2483	0.9667	1,375,911.2	5,433,111.0	795,088.3	2,857,213.0	0.55	0.94	739.0	794.0
	1.8	0.0533	0.1500	281,557.9	819,672.0	65,742.3	179,209.0	0.29	0.77	582.1	629.0
60	1	129.7750	180.1167	494,750,455.8	722,000,000.0	277,868,019.3	600,361,763.0	0.64	0.95	1,221.0	1,345.0
	1.4	8.8383	70.2500	43,867,673.7	354,662,071.0	40,753,595.6	343,376,419.0	0.69	0.97	882.1	951.0
	1.8	0.2967	0.7833	970,855.3	2,507,737.0	485,620.0	1,999,858.0	0.42	0.80	693.7	756.0

Table 5.6 Results for L/ A_ℓ =30/1, FR=0.5

		CPU Time (min.)		Total # of Nodes		# of Nodes Until Opt.		Opt. Nodes/Total Nodes		Total Cost(*1000)	
N	CT(*maxPT)	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	0.0000	0.0000	1,039.2	6,350.0	136.5	846.0	0.22	0.97	205.0	252.0
	1.4	0.0000	0.0000	406.4	1,127.0	77.6	421.0	0.21	0.50	152.9	193.0
	1.8	0.0000	0.0000	301.1	801.0	45.8	258.0	0.21	0.73	121.2	158.0
15	1	0.0000	0.0000	4,389.8	7,488.0	1,593.3	3,005.0	0.38	0.78	298.2	352.0
	1.4	0.0000	0.0000	3,082.2	7,780.0	1,348.1	7,761.0	0.29	1.00	221.2	265.0
	1.8	0.0000	0.0000	1,333.5	3,568.0	165.9	541.0	0.22	0.78	171.1	201.0
20	1	0.0133	0.0667	194,680.6	1,026,178.0	66,839.0	623,736.0	0.41	0.98	401.4	482.0
	1.4	0.0033	0.0167	21,029.1	92,951.0	15,820.5	88,485.0	0.55	0.98	289.2	336.0
	1.8	0.0000	0.0000	10,140.5	26,864.0	2,759.8	8,262.0	0.34	0.69	225.0	268.0
25	1	0.0900	0.6000	1,037,691.0	6,977,711.0	743,220.2	6,734,776.0	0.34	0.97	500.5	572.0
	1.4	0.0233	0.1167	250,929.4	1,365,984.0	48,920.2	204,612.0	0.39	0.78	365.6	406.0
	1.8	0.0033	0.0167	22,909.4	118,957.0	10,939.0	70,583.0	0.41	0.93	286.7	326.0
30	1	1.4767	8.8833	15,331,471.3	91,289,325.0	4,422,489.0	24,585,219.0	0.38	0.81	597.2	668.0
	1.4	0.1217	0.6833	1,340,270.7	8,282,708.0	405,407.2	1,529,079.0	0.42	0.66	431.4	468.0
	1.8	0.0583	0.3167	661,834.9	4,024,564.0	48,862.8	215,434.0	0.21	0.56	334.6	372.0
35	1	14.3300	70.4167	110,620,025.6	534,493,113.0	66,127,963.5	331,031,182.0	0.59	0.97	708.5	803.0
	1.4	0.3533	0.9500	3,096,194.3	8,306,088.0	1,636,761.7	5,497,210.0	0.54	0.95	511.6	569.0
	1.8	0.2500	1.1833	2,246,133.8	11,826,921.0	1,336,189.2	11,000,486.0	0.40	0.93	405.1	451.0
40	1	94.2800	180.0000	760,687,943.4	1,551,500,001.0	327,471,058.5	1,172,699,111.0	0.48	0.87	803.9	934.0
	1.4	14.6100	107.5500	131,346,883.7	1,028,215,627.0	21,589,444.1	79,525,808.0	0.65	0.90	575.6	665.0
	1.8	3.9700	27.3667	28,530,502.4	207,588,522.0	4,479,934.9	15,360,961.0	0.48	0.93	451.8	517.0
50	1	154.5050	180.6500	1,104,921,038.5	1,623,500,001.0	537,308,991.7	1,067,322,309.0	0.51	0.98	1,000.3	1,085.0
	1.4	59.3033	180.0000	359,904,199.9	1,213,000,000.0	90,122,467.3	303,462,354.0	0.45	0.87	718.5	787.0
	1.8	12.1783	40.2000	70,169,983.0	218,215,799.0	18,212,372.0	65,379,474.0	0.18	0.51	567.3	610.0

Table 5.7 Results for L/A_ℓ =30/1, FR=0.8

		CPU Time (min.)		Total # of Nodes		# of Nodes Until Opt.		Opt. Nodes/Total Nodes		Total Cost(*1000)	
N	CT(*maxPT)	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	0.0000	0.0000	28,543.8	111,231.0	1,980.9	9,427.0	0.17	0.81	203.4	252.0
	1.4	0.0000	0.0000	14,222.8	54,869.0	983.8	4,811.0	0.20	0.60	150.8	189.0
	1.8	0.0000	0.0000	7,238.9	29,417.0	226.1	715.0	0.08	0.28	120.2	158.0
15	1	0.0783	0.4333	1,267,329.2	7,220,749.0	181,841.0	785,157.0	0.23	0.85	292.8	347.0
	1.4	0.0217	0.1167	333,517.7	1,418,825.0	89,485.8	269,955.0	0.44	0.91	212.0	258.0
	1.8	0.0067	0.0167	107,638.9	282,695.0	19,652.1	113,683.0	0.19	0.59	168.9	196.0
20	1	8.9648	74.4300	145,351,071.8	1,200,009,626.0	4,809,865.4	34,689,878.0	0.24	0.81	382.7	447.0
	1.4	0.3704	1.6700	6,205,074.7	29,603,011.0	380,532.6	1,151,120.0	0.21	0.66	276.7	319.0
	1.8	0.1926	0.6000	2,980,171.3	10,345,436.0	240,909.7	1,063,567.0	0.18	0.70	216.7	256.0
25	1	51.8183	180.0000	756,441,136.7	2,825,496,557.0	400,292,436.9	2,004,678,496.0	0.48	0.93	480.7	540.0
	1.4	11.4350	57.1167	167,033,984.0	860,009,530.0	60,209,602.9	435,405,314.0	0.28	0.57	349.8	384.0
	1.8	3.5367	9.4667	48,395,857.1	133,458,170.0	14,708,283.4	49,983,646.0	0.44	0.91	271.9	294.0
30	1	180.0000	180.0000	2,442,830,682.5	2,741,607,198.0	1,390,490,391.1	2,268,631,859.0	0.58	0.99	583.0	660.0
	1.4	151.8100	180.0167	2,072,522,011.4	2,828,895,484.0	917,672,195.7	2,406,080,416.0	0.44	1.00	414.1	456.0
	1.8	115.0800	180.0000	1,534,588,888.6	2,557,657,875.0	532,239,919.0	1,468,879,666.0	0.37	0.85	321.1	355.0
20	1	26.0683	180.0000	417,883,451.3	2,870,674,867.0	5,024,385.9	34,689,878.0	0.21	0.81	387.8	447.0
	1.4	2.1667	18.3333	37,873,183.7	322,886,165.0	391,449.8	1,151,120.0	0.19	0.66	280.0	319.0
	1.8	8.8200	86.4667	172,911,469.9	1,702,293,157.0	216,818.7	1,063,567.0	0.16	0.70	219.8	256.0

balance can be achieved with fewer stations in more flexible lines, increasing the FR decreases the total cost.

Because ALB problems are NP hard, a very long CPU time is required to find the optimal solution for more flexible lines with large number of tasks. Therefore, as the problem size increases, the three-hour limit starts to take effect and the run terminates itself after 180 minutes with the best solution found until this time. Table 5.8 gives the values of parameters for which the optimal solution is found within three hours for all ten problems.

Table 5.8 Experimental conditions for which the optimal solution is found within three hours

L/A_ℓ	FR	N	CT (*maxPT)
30/30	0.2	70	1.4, 1.8
	0.5	50	1.8
	0.8	30	1.8
30/1	0.2	60	1.4, 1.8
	0.5	50	1.8
	0.8	25	1.4, 1.8

Opt.Nodes/Total Nodes ratio in Tables 5.2-5.7 indicates how early in the search process the optimal solution is found. When L/A_ℓ is 30/30 and FR is 0.2, maximum ratio is 1.00 and maximum average ratio is 0.33. This implies that among the problems solved under this combination, optimal solution is found after searching at most 33% of the branch and bound tree on the average. Maximum ratio and maximum average ratio are 0.88 and 0.38 for FR 0.5, 0.97 and 0.43 for FR 0.8.

When L/A_ℓ is 30/1 and FR is 0.2, maximum ratio is 1.00 and maximum average ratio is 0.69 which implies that an optimal solution is found after searching at most 69% of the branch and bound tree on the average. When FRs are 0.5 and 0.8, the maximum ratio and maximum average ratio are 1.00 and 0.65, 1.00 and 0.58, respectively.

If the tables are carefully examined it is seen that most of the average ratios are smaller than 0.50. This means that the optimal solution for most of the problems can be found before 50% of the branch and bound tree is searched on the average. Hence, we can say that the depth-first search with complete branching strategy well fits our problem. Because of this we believe that our algorithm is capable of finding optimal solutions within three hours even for larger problems.

5.4.2 Comparison with Askin and Zhou' s Heuristic

As far as the total cost is concerned, the branch and bound algorithm brings a significant improvement over the heuristic of Askin and Zhou (1997) for most parameter values. The percentage deviations in total cost by the branch and bound algorithm compared to their heuristic are given in Tables 5.9 and 5.10 when L/A_ℓ is 30/30 and 30/1.

When L/A_ℓ is 30/30, reduction in total cost by the branch and bound algorithm is 2-13%, 4-12%, 6-23% for FRs 0.2, 0.5 and 0.8. The same values are 1-5%, 3-6%, 4-11% when L/A_ℓ is 30/1. It seems Askin and Zhou' s heuristic does relatively better with low FRs and low equipment cost (when L/A_ℓ is 30/1). Using the exact branch and bound procedure brings more substantial savings when flexibility is high and when equipment cost is comparable to station opening cost.

Maximum improvement over the heuristic of Askin and Zhou (1997) varies in ranges 10-30%, 8-31% and 14-57% with three FRs when L/A_ℓ is 30/30. When L/A_ℓ is 30/1, the same ranges are 2-14%, 7-22% and 10-24%. It seems their

Table 5.9 Results of Percentage Deviation for L/ A_ℓ =30/30

		% Deviation for FR=0.2		% Deviation for FR=0.5		% Deviation for FR=0.8	
N	CT(*maxPT)	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	5	18	7	17	6	20
	1.4	5	10	8	29	13	20
	1.8	3	15	7	27	20	28
15	1	9	30	7	15	14	20
	1.4	11	24	11	31	15	31
	1.8	7	29	6	14	22	40
20	1	7	14	6	14	11	19
	1.4	13	30	12	30	20	40
	1.8	3	13	11	24	21	35
25	1	5	18	7	17	6	20
	1.4	5	10	8	29	13	20
	1.8	3	15	7	27	20	28
30	1	8	14	6	16	7	14
	1.4	6	18	8	21	23	39
	1.8	3	12	10	18	32	57
35	1	6	13	7	11		
	1.4	7	13	7	12		
	1.8	5	13	5	10		
40	1	9	21	5	12		
	1.4	7	15	10	22		
	1.8	2	7	6	11		
50	1	8	15	4	11		
	1.4	8	13	6	14		
	1.8	2	6	5	8		
60	1	8	14				
	1.4	6	15				
	1.8	2	7				
70	1	7	12				
	1.4	6	11				
	1.8	3	7				

		% Deviation	
N&FR	CT(*maxPT)	Avg.	Max.
20	1	10	19
0.8	1.4	21	40
	1.8	20	35
40	1	9	21
0.2			
70	1.4	7	11
0.2	1.8	4	9

Table 5.10 Results of Percentage Deviation for L/ A_ℓ =30/1

		% Deviation for FR=0.2		% Deviation for FR=0.5		% Deviation for FR=0.8	
N	CT(*maxPT)	Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	4	14	4	11	6	12
	1.4	4	11	3	10	7	13
	1.8	5	11	5	12	9	15
15	1	5	12	5	13	6	12
	1.4	4	14	5	16	10	16
	1.8	1	2	5	22	5	24
20	1	4	9	4	10	5	10
	1.4	3	9	5	10	7	16
	1.8	3	14	6	15	6	16
25	1	4	14	4	11	6	12
	1.4	4	11	3	10	7	13
	1.8	5	11	5	12	9	15
30	1	4	9	3	7	4	7
	1.4	3	7	5	9	7	11
	1.8	5	11	6	13	11	14
35	1	6	11	5	12		
	1.4	1	3	4	8		
	1.8	4	10	5	10		
40	1	3	7	5	11		
	1.4	5	10	5	8		
	1.8	3	10	4	10		
50	1	4	7	4	7		
	1.4	2	5	4	7		
	1.8	4	6	5	10		
60	1	4	7				
	1.4	4	7				
	1.8	4	7				

		% Deviation	
N&FR	CT(*maxPT)	Avg.	Max.
20	1	5	10
0.8	1.4	7	16
	1.8	5	16

heuristic also becomes less consistent in solution quality as FR increases and as L/A_ℓ is changed from 30/1 to 30/30. Detailed information about frequency distributions is given in Appendix B.

The other two parameters, N and CT, do not seem to have any systematic effects on the solution quality.

When branch and bound algorithm and their heuristic are compared in terms of the CPU time, the heuristic procedure finds a solution in less than 1 second in all problems for each combination of parameters. The CPU time of branch and bound algorithm is measured in minutes and increases drastically as the problem size becomes larger. For more flexible lines, the gap between computation times of the heuristic procedure and the branch and bound procedure is also very large. The heuristic clearly outperforms the branch and bound algorithm in terms of computation time.

5.4.3 Station Paralleling

Tables 5.11-5.12 show the average and maximum number of stations opened in the branch and bound solutions. Changes in cost structure affect the number of stations. When L/A_ℓ is 30/30, the number of parallel stations is rather small. When L/A_ℓ is 30/1, more stations are paralleled.

From these tables, we can draw a conclusion about the effect of cost structure on the number of stations and station paralleling. When equipment cost is much lower relative to station opening cost (i.e. L/A_ℓ is 30/1), paralleling of stations does not cause considerable increase in cost due to equipment duplication. In this case more stations are paralleled and a balance with fewer stations is obtained. However, when equipment cost is comparable to station opening cost (i.e. L/A_ℓ is 30/30), a longer line with larger number of serial stations is obtained.

Table 5.11 Number of Stations When L/A_ℓ =30/30

N	CT(*maxPT)	Level of Paralleling	# of Stations for FR=0.2		# of Stations for FR=0.5		# of Stations for FR=0.8	
			Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	1	6.8	10	6.3	9	5.6	9
		2	0.1	1	0.2	1	0.0	0
		3	0.2	1	0.2	1	0.4	2
	1.4	1	4.8	8	5	6	4.0	5
		2	0.2	1	0.3	1	0.5	1
		3	0.1	1	0	0	0.0	0
	1.8	1	4.3	5	3.8	5	4.0	5
		2	0	0	0.1	1	0.0	0
		3	0	0	0	0	0.0	0
15	1	1	9.9	15	9.2	13	8.9	13
		2	0.6	2	0.5	2	0.5	2
		3	0	0	0.1	1	0.1	1
	1.4	1	6.5	9	6.9	9	6.8	9
		2	0.6	2	0.4	1	0.3	2
		3	0.1	1	0	0	0.0	0
	1.8	1	5.6	7	5.6	7	5.1	7
		2	0.2	1	0	0	0.2	1
		3	0	0	0	0	0.0	0
20	1	1	15.1	19	13.3	18	11.4	17
		2	0.1	1	0.1	1	0.3	1
		3	0.1	1	0.2	2	0.3	2
	1.4	1	8.6	12	8	12	8.7	11
		2	0.4	3	0.9	3	0.3	2
		3	0.4	2	0.1	1	0.1	1
	1.8	1	6.7	10	7.7	10	6.4	9
		2	0.3	2	0	0	0.1	1
		3	0.2	1	0	0	0.1	1
25	1	1	18.2	22	17.2	20	16.5	19
		2	0.2	2	0.3	3	0.2	2
		3	0	0	0	0	0	0
	1.4	1	12.9	16	12.7	15	12	14
		2	0.2	1	0.3	2	0	0
		3	0.1	1	0	0	0	0
	1.8	1	8.9	11	9.6	12	8.9	10
		2	0.1	1	0.1	1	0.1	1
		3	0.2	2	0	0	0	0
30	1	1	22.3	25	21.2	25	17.9	23
		2	0.2	2	0.3	3	0.9	3
		3	0	0	0	0	0.1	1
	1.4	1	15.4	19	14.1	17	12.7	15
		2	0.3	1	0.7	3	0.8	2
		3	0.1	1	0	0	0	0
	1.8	1	11.3	14	11.1	12	10.7	12
		2	0.3	2	0.1	1	0.1	1
		3	0	0	0	0	0	0
35	1	1	26.2	30	25.7	31		
		2	0.1	1	0.1	1		
		3	0	0	0	0		
	1.4	1	19.7	24	18.3	21		
		2	0.1	1	0.1	1		
		3	0	0	0	0		
	1.8	1	14	16	13.3	15		
		2	0	0	0.1	1		
		3	0	0	0	0		

Table 5.11 Number of Stations When L/A$_\ell$=30/30 (Cont.)

N	CT(*maxPT)	Level of Paralleling	# of Stations for FR=0.2 Avg.	# of Stations for FR=0.2 Max.	# of Stations for FR=0.5 Avg.	# of Stations for FR=0.5 Max.
40	1	1	30.6	37	27.3	35
		2	0.2	1	0.2	1
		3	0	0	0.3	3
	1.4	1	20.5	27	19.1	26
		2	0.2	1	0.7	4
		3	0.3	3	0.1	1
	1.8	1	15	18	14.4	17
		2	0.1	1	0.2	1
		3	0.1	1	0.1	1
50	1	1	37.6	41	35.4	40
		2	0	0	0.2	1
		3	0	0	0	0
	1.4	1	27.5	31	25.6	29
		2	0.1	1	0.1	1
		3	0	0	0	0
	1.8	1	19.5	22	18.6	21
		2	0	0	0	0
		3	0	0	0	0
60	1	1	44.6	48		
		2	0	0		
		3	0	0		
	1.4	1	32.7	36		
		2	0.1	1		
		3	0	0		
	1.8	1	23	25		
		2	0.1	1		
		3	0	0		
70	1	1	50.8	57		
		2	0.5	5		
		3	0.1	1		
	1.4	1	37.4	41		
		2	0.1	1		
		3	0	0		
	1.8	1	26.9	29		
		2	0	0		
		3	0	0		

N&FR	CT(*maxPT)	Level of Paralleling	# of Stations Avg.	# of Stations Max.
20	1	1	10.8	17
0.8		2	0.6	3
		3	0.4	2
	1.4	1	8.1	11
		2	0.5	2
		3	0.2	1
	1.8	1	6.6	9
		2	0.1	1
		3	0.1	1
40	1	1	28.9	37
0.2		2	0.5	3
		3	0.2	2
70	1.4	1	35.7	41
0.2		2	0.6	5
		3	0.1	1
	1.8	1	26.1	29
		2	0.3	3
		3	0	0

Table 5.12 Number of Stations When L/A_ℓ =30/1

N	CT(*maxPT)	Level of Paralleling	# of Stations for FR=0.2		# of Stations for FR=0.5		# of Stations for FR=0.8	
			Avg.	Max.	Avg.	Max.	Avg.	Max.
10	1	1	3.5	6	3.9	6	4.5	8
		2	0.7	2	0.7	2	0.1	1
		3	0.6	1	0.4	1	0.6	2
	1.4	1	1.9	3	1.9	3	3.0	5
		2	0.7	2	1.3	2	0.9	2
		3	0.5	1	0.1	1	0.0	0
	1.8	1	2	5	2.7	5	3.4	5
		2	0.6	2	0.4	1	0.2	1
		3	0.2	1	0.1	1	0.0	0
15	1	1	4.3	10	4.4	7	5.3	7
		2	1.1	3	1.1	2	1.4	3
		3	1	2	0.9	2	0.4	1
	1.4	1	3	4	3.2	5	3.5	6
		2	1.5	2	1.2	2	1.1	3
		3	0.3	1	0.4	1	0.3	2
	1.8	1	3.6	6	4.3	6	4.3	6
		2	0.9	2	0.5	2	0.5	2
		3	0	0	0	0	0.0	0
20	1	1	6.1	12	7.5	13	7.9	11
		2	2.2	5	1.5	4	1.1	2
		3	0.8	1	0.7	2	0.7	3
	1.4	1	3.8	7	3.4	6	5.6	9
		2	1.3	3	1.7	4	1.2	3
		3	0.9	2	0.7	2	0.2	1
	1.8	1	2.6	4	4.7	8	6.2	8
		2	1.4	3	0.8	3	0.1	1
		3	0.5	1	0.2	1	0.1	1
25	1	1	7.7	19	7.6	15	7.5	14
		2	2.5	5	2.7	5	2.4	5
		3	1	3	0.8	3	0.9	3
	1.4	1	4.6	8	5.1	9	8.1	10
		2	2.3	4	2.4	5	1.4	3
		3	0.6	2	0.4	2	0	0
	1.8	1	4.3	10	5.5	9	7.7	9
		2	1.3	3	0.8	3	0.2	1
		3	0.6	2	0.5	2	0.1	1
30	1	1	9.3	19	9.6	16	9.6	15
		2	2.8	5	3	6	2.4	5
		3	1.4	4	1	2	1.3	3
	1.4	1	4.6	8	3.7	7	9.6	13
		2	2.7	5	3.6	5	1.4	3
		3	1.1	2	0.7	2	0.2	1
	1.8	1	4.7	9	4.4	7	9.4	11
		2	2.1	4	2.2	3	0.3	1
		3	0.5	2	0.4	1	0	0

Table 5.12 Number of Stations When L/A_ℓ =30/1 (Cont.)

N	CT(*maxPT)	Level of Paralleling	# of Stations for FR=0.2		# of Stations for FR=0.5	
			Avg.	Max.	Avg.	Max.
35	1	1	9.3	19	9.3	11
		2	3.3	7	4.3	7
		3	2.1	5	1.3	3
	1.4	1	6.8	11	8.5	13
		2	3.6	6	3.1	4
		3	0.6	1	0.3	2
	1.8	1	7.5	13	7.5	11
		2	2.1	4	2	6
		3	0.2	1	0.2	1
40	1	1	9.5	19	10.5	16
		2	4.9	7	4.9	8
		3	2	5	1.5	4
	1.4	1	5.3	10	7.8	13
		2	3.9	5	4.1	6
		3	1.5	3	0.5	2
	1.8	1	7	12	7.5	12
		2	2.5	4	2.4	5
		3	0.6	2	0.4	1
50	1	1	13.1	20	14.6	20
		2	6	8	5.2	8
		3	2	5	1.9	4
	1.4	1	6.5	12	9.5	14
		2	6.2	8	5	7
		3	1	2	0.7	2
	1.8	1	7.2	14	10.9	17
		2	4	7	2.7	5
		3	0.6	2	0.2	2
60	1	1	16.9	27		
		2	5.8	8		
		3	2.9	6		
	1.4	1	6.7	11		
		2	7	9		
		3	1.7	3		
	1.8	1	10	16		
		2	4.5	6		
		3	0.4	1		

N&FR	CT(*maxPT)	Level of Paralleling	# of Stations	
			Avg.	Max.
20	1	1	7.6	11
0.8		2	1.3	3
		3	0.7	3
	1.4	1	5.3	9
		2	1.3	3
		3	0.3	1
	1.8	1	6.2	8
		2	0.2	1
		3	0.1	1

CHAPTER 6

CONCLUSIONS

In this study, we develop an exact algorithm for single model deterministic ALB problem with station paralleling. We aim to minimize a function of number of workstations and total equipment cost. We propose a branch and bound algorithm that allows arbitrary level of paralleling, of each stage. We propose some lower and upper bounding mechanisms to enhance the efficiency of the algorithm. To the best our knowledge, our study is the first optimization attempt to this problem.

We conduct a computational study to see the effects of some parameters on the performance measures. We examine the effects of problem size, cycle time, cost structure and Flexibility ratio on the CPU time and the total number of nodes generated. The performance of the Askin and Zhou (1997) heuristic solution as a percentage deviation from the optimal solution is also analyzed. The main conclusions can be summarized as follows:

For all performance measures, the most significant parameter is the Flexibility ratio,FR. Increasing FR value while keeping the other parameters constant, increases the total number of nodes and CPU time exponentially. The exponential nature of increase becomes clearer for problem sizes greater than or equal to 20. As a balance can be achieved with fewer stations in more flexible lines, increasing the FR decreases the total cost.

Because our problem is NP hard, it requires long time to reach the optimal solution. For each problem we set maximum run time as three hours. The run

terminates itself after three hours with the best solution found until this time. For different FR and L/A_ℓ values we determine the limit on problem size that the optimal solution is found within three hours. When FR value is increased, it is seen that problem size for which the optimal solution is found within three hours decreases.

When Opt.Nodes/Total Nodes ratio is considered, it is seen that for each combination of parameter values, most of the average ratios are smaller than 0.50. This indicates that for most of the problems, the optimal solution can be found after searching only half of the branch and bound tree on the average. So, we can say that the depth-first search with complete branching strategy well fits our problem, and our lower bounds are good estimates of the optimal cost.

When we compare the heuristic of Askin and Zhou (1997) with our branch and bound algorithm, we see that using the exact procedure brings more substantial savings when flexibility is high and when equipment cost is comparable to station opening cost. Also, the heuristic becomes less consistent in solution quality as FR increases and as L/A_ℓ is changed from 30/1 to 30/30.

If we examine the effect of cost structure on the number of stations and station paralleling, it is seen that when equipment cost is much lower relative to station opening cost, more stations are paralleled and a balance with fewer stations is obtained. The reason for this can be explained as the paralleling of stations does not cause considerable increase in cost due to equipment duplication. But when equipment cost is comparable to station opening cost, a longer line with larger number of serial stations is obtained.

Our algorithm can also be used in mixed model assembly line balancing after forming the combined precedence diagram.

Although there are two types of paralleling, i.e. task paralleling and station paralleling, there is not any comparison studies between these types in literature. A

comparison study between these types of paralleling can be made in terms of cost optimization.

Further research directions might include:

- Heuristic procedures that are capable of solving problems of larger sizes, for example metaheuristics.
- Generation of all efficient solutions relative to number of workstations and total cost.
- Stochastic task times and/or cycle time.

REFERENCES

Arcus, A.L., 1966. 'COMSOAL: A computer method of sequencing operations for assembly lines', *International Journal of Production Research*, **4,** 259-277.

Askin, R. G. and Standridge, C. H., 1993. *Modelling and Analysis of Manufacturing Systems*, John Wiley & Sons, New York.

Askin, R. G. and Zhou, M., 1997. 'A parallel station heuristic for the mixed model production line balancing problem', *International Journal of Production Research*, **35** (11), 3095-3105.

Bard, J. F., 1989. 'Assembly line balancing with parallel workstations and dead time', *International Journal of Production Research*, **27** (6), 1005-1018.

Baybars, I., 1986. 'A survey of exact algorithms for the simple assembly line balancing problem', *Management Science*, **32** (8), 909-932.

Berger, I., Bourjolly, J. M. and Laporte, G., 1992. 'Branch-and-bound algorithms for the multi-product assembly line balancing problem', *European Journal of Operational Research*, **58**, 215-222.

Buxey, G. M., 1974. 'Assembly line balancing with multiple stations', *Management Science*, **20** (6), 1010-1021.

Chakravarty, A. K. and Shtub, A., 1985. 'Balancing mixed model lines with in-process inventories', *Management Science*, **31** (9), 1161-1174.

Erel, E. and Sarin, S. C., 1998. 'A survey of the assembly line balancing procedures', *Production Planning & Control*, **9** (5), 414-434.

Fokkert, J. I. Z. and de Kok, T.G., 1997. 'The mixed and multi-model line balancing problem: A comparison', *European Journal of Operational Research*, **100**, 399-412.

Ghosh, S., and Gagnon, R. J., 1989. 'A comprehensive literature review and analysis of the design, balancing and scheduling of the assembly systems', *International Journal of Management Science*, **27** (4), 637-670.

Gökçen, H., 1997. 'Montaj hattı etkinliğinin parallel istasyonlarla artırılması: Bir sezgisel yöntem', *Gazi Univ. Fen Bil. Enst. Dergisi*, **10** (3), 433-442.

Gökçen, H. and Erel, E., 1997. 'Binary integer formulation for mixed-model assembly line balancing problem', *Computers and Industrial Engineering*, **34** (2), 451-461.

Hackman, S. T., Magazine, M. J. and Wee, T. S., 1989. 'Fast, effective algorithms for simple assembly line balancing problems', *Operations Research*, **37** (6), 916-924.

Hax, A. C. and Candea, D., 1984. *Production and Inventory Management*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Helgeson, W. P. and Birnie, D. P., 1961. 'Assembly line balancing using the ranked positional weight technique', *Journal of Industrial Engineering*, **12** (6), 394-398.

Hoffman, T. R., 1963. 'Assembly line balancing with a precedence matrix', *Management Science*, **9**, 551-562.

Hoffman, T. R., 1992. 'EUREKA: A hybrid system for assembly line balancing', *Management Science*, **38**, 39-47.

Jackson, J. R., 1956. 'A computing procedure for a line balancing problem', *Management Science*, **2**, 261-271.

Johnson, R. V., 1988. 'Optimally balancing large assembly lines with FABLE', *Management Science*, **34** (2), 240-253.

Kilbridge, M. D. and Wester, L., 1961. 'A heuristic method of assembly line balancing', *Journal of Industrial Engineering*, **12**, 292-298.

Klein, R. and Scholl, A., 1996. 'Maximizing the production rate in simple assembly line balancing – A branch-and-bound procedure', *European Journal of Operational Research*, **62**, 367-385.

Macaskill, J. L. C., 1972. 'Production line balances for mixed model lines', *Management Science*, **19** (4), 423-434.

McMullen, P. R. and Frazier, G. U., 1998. 'Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations', *International Journal of Production Research*, **36** (10), 2717-2741.

Miltenburg, G. J. and Wijngaard, J., 1994. 'The U-line balancing problem', *Management Science*, **40** (10), 1378-1388.

Monden, Y., 1983. *Toyota Production System*, Institute of Industrial Engineers, Atlanta.

Moodie, C. L. and Young, H. H., 1965. 'A heuristic method of assembly line balancing for assumptions of constant or variable work element times', *Journal of Industrial Engineering*, **16**, 23-29.

Patterson, J. H. and Albracht, J. J., 1975. 'Assembly Line Balancing: 0-1 programming with Fibonacci Search', *Operations Research*, **23**, 166-174.

Ponnambalom, S. G., Aravindan, P., and Naidu, G. M., 1999. 'A comperative evaluation of assembly line balancing heuristics', *International Journal of Advanced Manufacturing Technology*, **15**, 577-586.

Salveson, M. E., 1955. 'The assembly line balancing problem', *Journal of Industrial Engineering*, **6**, 18-25.

Sarker, B. R. and Shanthikumar, J. G., 1983. 'A generalized approach for serial or parallel line balancing', *International Journal of Production Research*, **21** (1), 109-133.

Scholl, A. and Klein, R., 1997. 'SALOME: A bidirectional branch-and-bound procedure for assembly line balancing', *INFORMS Journal on Computing*, **9**, 319-334.

Scholl, A. and Klein, R., 1999. 'Balancing assembly lines effectively-A computational comparison', *European Journal of Operational Research*, **114**, 50-58.

Sprecher, A., 1999. 'A competitive branch-and-bound algorithm for the simple assembly line balancing problem', *International Journal of Production Research*, **37** (8), 1787-1816.

Pinto, P., Dannenbring, D. G. and Khumawala, B. M., 1975. 'A branch and bound algorithm for assembly line balancing with paralleling', *International Journal of Production Research*, **13** (2), 183-196.

Pinto, P., Dannenbring, D. G. and Khumawala, B. M., 1981. 'Branch and bound and heuristic procedures for assembly line balancing with paralleling of stations', *International Journal of Production Research*, **19** (5), 565-576.

Talbot, F. B. and Patterson, J. H., 1984. 'An integer programming algorithm with network cuts for solving the single model assembly line balancing problem', *Management Science*, **30**, 85-99.

Thangavelu, S. R. and Shetty, C. M., 1971. 'Assembly line balancing by zero-one integer programming', *AIIE Trans.*, **3**, 61-68.

Thomopoulos, N. T., 1967. 'Line balancing-sequencing for mixed model assembly', *Management Science*, **14** (2), 59-75.

Uğurdağ, H. F., Rachmadugu, R. and Papachristou, C. A., 1997. 'Designing paced assembly lines with fixed number of stations', *European Journal of Operational Research*, **102**, 488-501.

Wild, R., 1972. *Mass-production Management. The Design and Operation of Production Flow-Line Systems,* John Wiley and Sons, New York.

APPENDIX A

SOLUTIONS TO TEST DATA

Table A.1 Optimal Solutions of Test Problems

Name	c (given)	Minimal number of stations	Name	c (given)	Minimal number of stations
Bowman	20	5	**Sawyer**	36	10
Mansoor	48	4	**Sawyer**	41	8
Mansoor	62	3	**Sawyer**	54	7
Mansoor	94	2	**Sawyer**	75	5
Mertens	6	6	**Kilbridge**	138	4
Mertens	7	5	**Kilbridge**	184	3
Mertens	8	5	**Roszieg**	14	10
Mertens	10	3	**Roszieg**	16	8
Mertens	15	2	**Roszieg**	18	8
Mertens	18	2	**Roszieg**	21	6
Jaeschke	6	8	**Roszieg**	25	6
Jaeschke	7	7	**Roszieg**	32	4
Jaeschke	8	6	**Buxey**	27	13
Jaeschke	10	4	**Buxey**	30	12
Jaeschke	18	3	**Buxey**	33	11
Jackson	7	8	**Buxey**	36	10
Jackson	9	6	**Buxey**	41	8
Jackson	10	5	**Buxey**	47	7
Jackson	13	4	**Buxey**	54	7
Jackson	14	4	**Lutz1**	1414	11
Jackson	21	3	**Lutz1**	1572	10
Mitchell	14	8	**Lutz1**	1768	9
Mitchell	15	8	**Lutz1**	2020	8
Mitchell	21	5	**Lutz1**	2357	7
Mitchell	26	5	**Lutz1**	2828	6
Mitchell	35	3	**Gunther**	41	14
Mitchell	39	3	**Gunther**	44	12
Heskiaoff	138	8	**Gunther**	49	11
Heskiaoff	205	5	**Gunther**	54	9
Heskiaoff	216	5	**Gunther**	61	9
Heskiaoff	256	4	**Gunther**	69	8
Heskiaoff	324	4	**Gunther**	81	7
Heskiaoff	342	3	**Hahn**	2004	8
Sawyer	25	14	**Hahn**	3507	5
Sawyer	27	13	**Hahn**	4676	4
Sawyer	30	12			

APPENDIX B

FREQUENCY DISTRIBUTIONS OF DEVIATION

Figures B.1 and B.2 show the frequency of average and maximum percentage deviation values when L/A_ℓ is 30/30 and FR is 0.2. As seen from Figure B.1, 60% of the observations are between 5 and 8. When maximum percentage deviation values are considered, it is seen that 53% of the observed values are between 13 and 18.

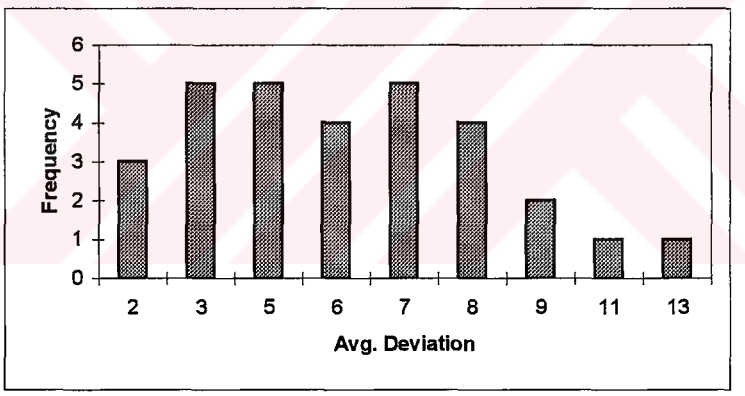


Figure B.1 Frequency of Avg. % Deviation When L/A_ℓ is 30/30 and FR is 0.2

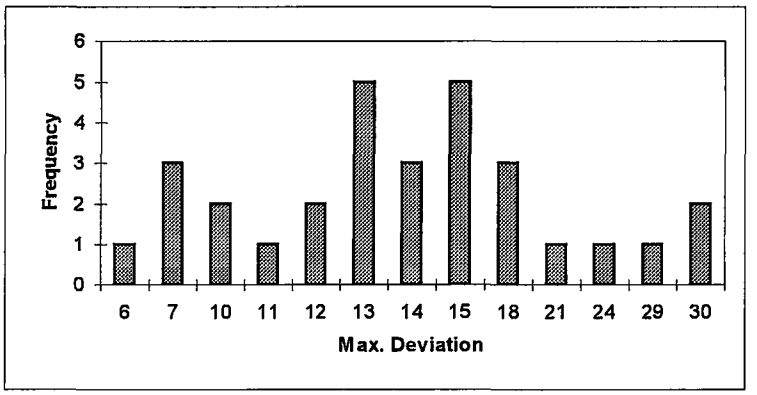


Figure B.2 Frequency of Max. % Deviation When L/A$_\ell$ is 30/30 and FR is 0.2

Figures B.3 and B.4 show the frequency of percentage deviation values when L/A is 30/30 and FR is 0.5. Average percentage deviation value of 7 has the highest frequency. 75% of the observations are between 5 and 8. There are various observed maximum percentage deviation values each with low frequency. It is difficult to categorize them. 38% of the values are between 21 and 31, and 33% of them are between 11 and 14.

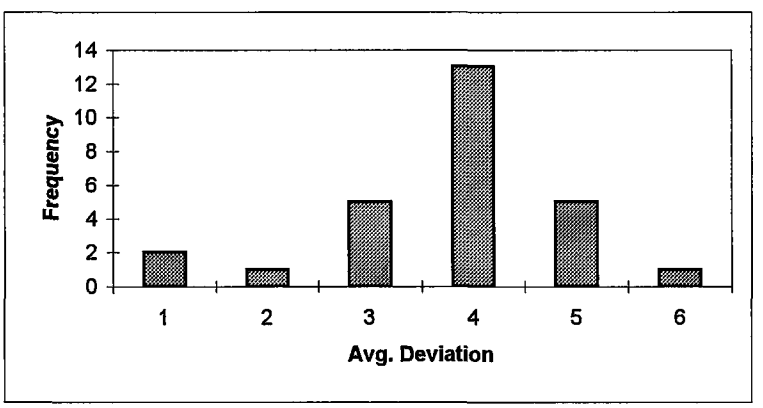


Figure B.3 Frequency of Avg. % Deviation When L/A$_\ell$ is 30/30 and FR is 0.5

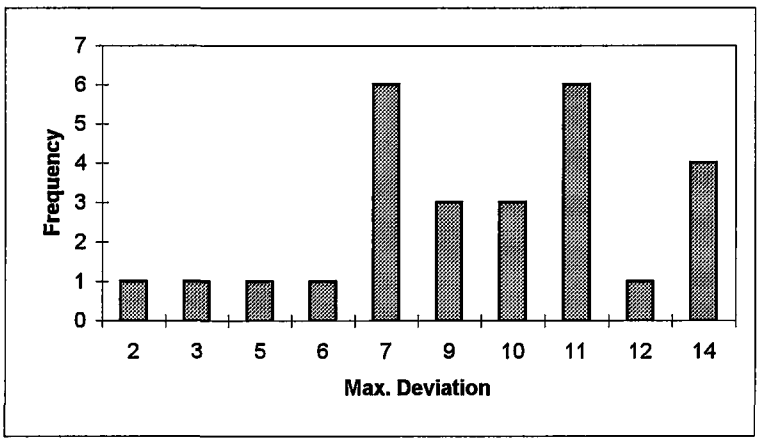


Figure B.4 Frequency of Max. % Deviation When L/A$_\ell$ is 30/30 and FR is 0.5

The frequency of average and maximum percentage deviation values when L/A$_\ell$ is 30/30 and FR is 0.8 are shown in Figures B.5 and B.6, respectively. There are various observed average values each with low frequency. 47% of them are between 20 and 32. The highest frequency maximum percentage deviation is 20 with a relative frequency of 33%. 80% of the values are between 20 and 40. The highest value observed during the analysis is 57.

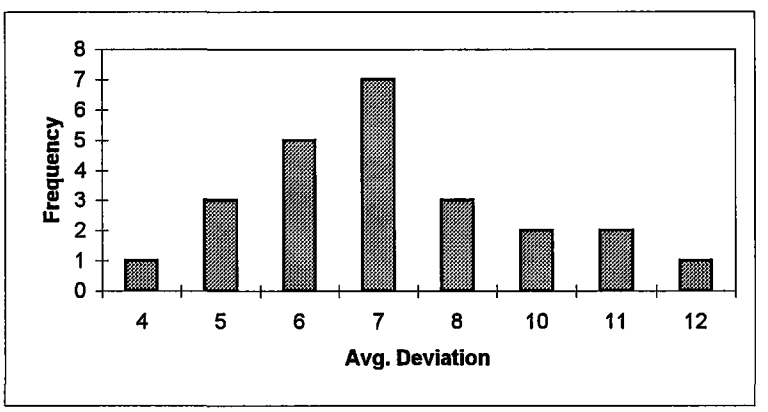


Figure B.5 Frequency of Avg. % Deviation When L/A$_\ell$ is 30/30 and FR is 0.8

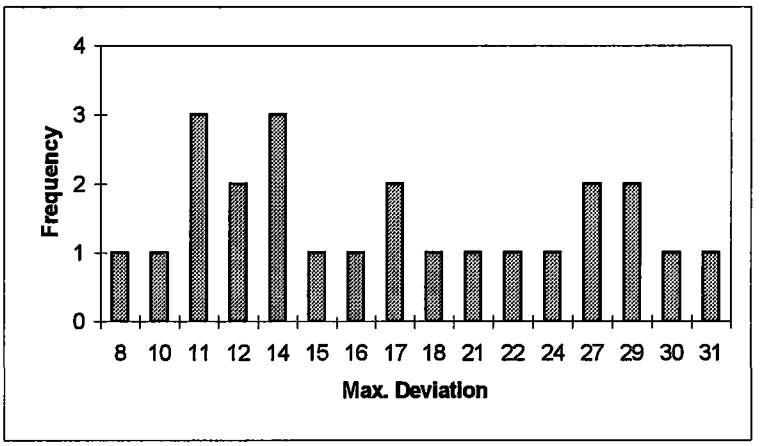


Figure B.6 Frequency of Max. % Deviation When L/A$_\ell$ is 30/30 and FR is 0.8

Figures B.7 and B.8 show the frequency of average and maximum percentage deviation values when L/A$_\ell$ is 30/1 and FR is 0.2. 48% of the average values equal to 4. 85% of the values are between 3 and 5. Maximum values of 7 and 11 are most frequently observed with 67%.

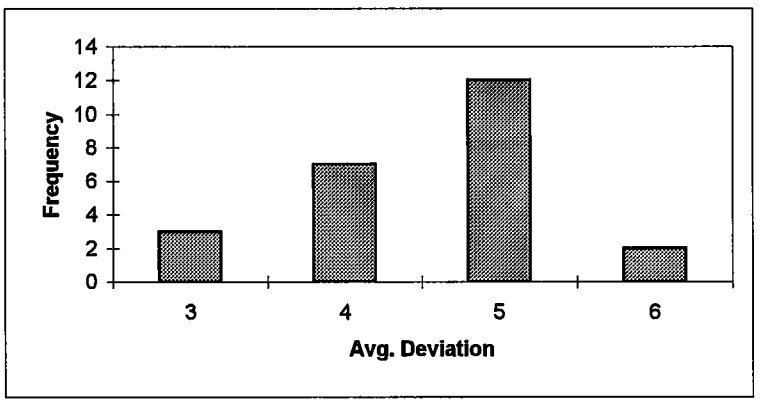


Figure B.7 Frequency of Avg. % Deviation When L/A$_\ell$ is 30/1 and FR is 0.2

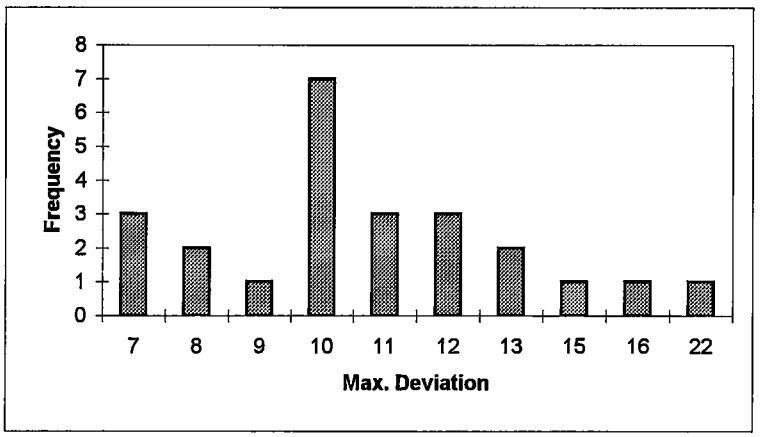


Figure B.8 Frequency of Max. % Deviation When L/A$_\ell$ is 30/1 and FR is 0.2

Figures B.9 and B.10 show the frequency of percentage deviation values when L/A$_\ell$ is 30/1 and FR is 0.5. Most frequently observed average value is 5. Relative frequency of this value is 50%. Maximum percentage deviation value 10 is observed with a relative frequency of 29%. 63% of the observations are between 0.10 and 0.13.

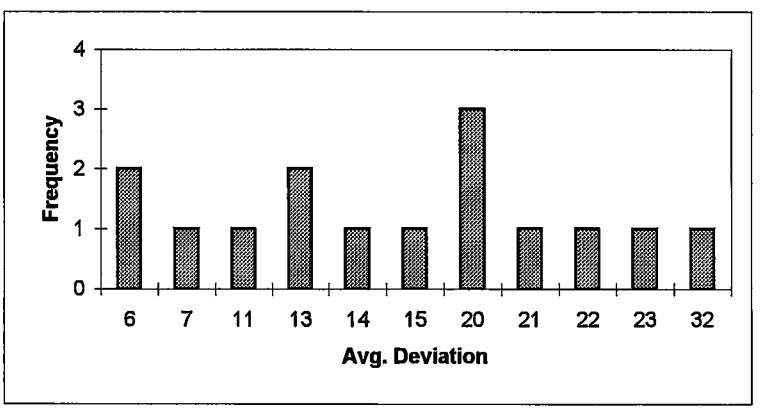


Figure B.9 Frequency of Avg. % Deviation When L/A$_\ell$ is 30/1 and FR is 0.5

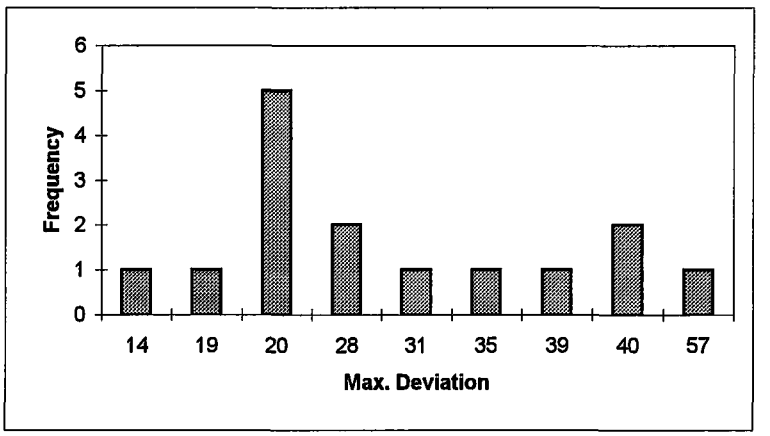


Figure B.10 Frequency of Max. % Deviation When L/A$_\ell$ is 30/1 and FR is 0.5

Figures B.11 and B.12 show the frequency of percentage deviation values when L/A$_\ell$ is 30/1 and FR is 0.8. Highest frequency average values are 6 and 7 each with a relative frequency of 27%. There are various maximum percentage deviation values each with low frequency. 60% of the observations are between 13 and 24.

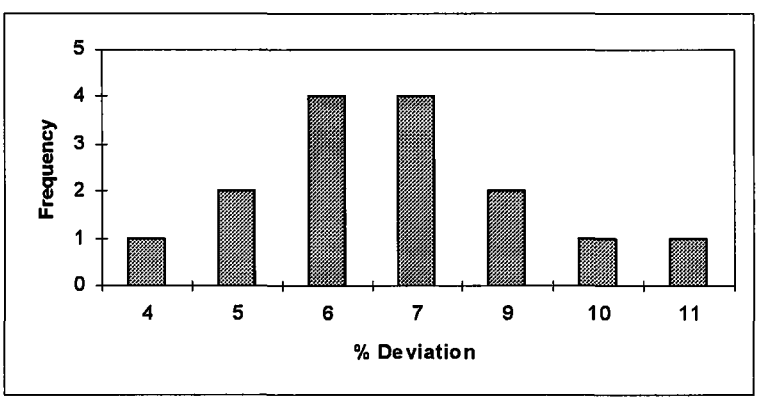


Figure B.11 Frequency of Avg. % Deviation When L/A$_\ell$ is 30/1 and FR is 0.8

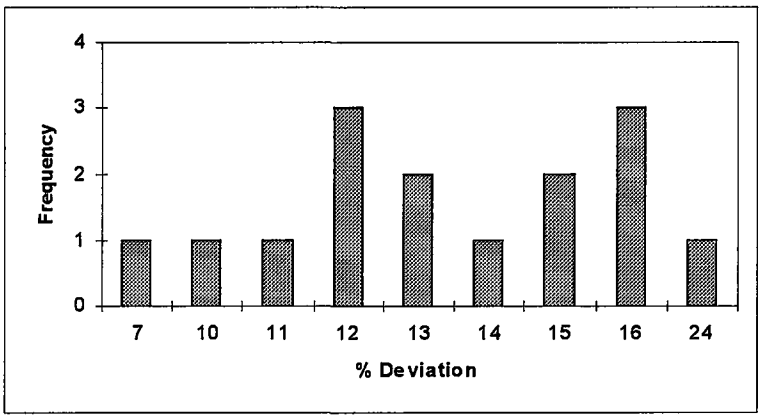


Figure B.12 Frequency of Max. % Deviation When L/A$_\ell$ is 30/1 and FR is 0.8

When L/A$_\ell$ is 30/30, average percentage deviation values are between 5 and 8 for FRs 0.2 and 0.5. However, when FR is 0.8, the average values are higher. Nearly all of the observed values are greater than 11. When maximum percentage deviation values are considered, the most frequently observed values also get larger as the FR increases. The difference is more obvious when FR is 0.8. When L/A$_\ell$ is 30/1, average percentage deviation values are generally between 4 and 7 for any FR. When we consider maximum percentage deviation, it is seen that the most frequently observed values get larger as the FR increases.