BALANCING STRAIGHT AND U-TYPE ASSEMBLY LINES WITH STOCHASTIC PROCESS TIMES

A THESIS
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL
ENGINEERING AND THE INSTITUTE OF ENGINEERING AND
SCIENCE OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Halil Şekerci
August, 2003

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

Prof. İhsan Sabuncuoğlu (Principal Advisor)

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

Prof. Erdal Erel

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

Asst. Prof. M. Murat Fadıloğlu

Approved for the Institute of Engineering and Science:

Prof. Mehmet Baray
Director of Institute of Engineering and Science

ABSTRACT

BALANCING STRAIGHT AND U-TYPE ASSEMBLY LINES WITH STOCHASTIC PROCESS TIMES

Halil Şekerci
M.S. in Industrial Engineering
Advisor: Prof. İhsan Sabuncuoğlu
August,2003

In this thesis, we study the problem of assembly line balancing with stochastic task process times. The research considers both the well-known straight line balancing problem and U-line balancing problem where the line is paced, with no buffer inventories between stations. The objective is to minimize a two component cost function where the cost terms come from cost of manning the line and cost of finishing the incomplete units off the line. Cost is measured by an existing exact method for straight line balancing and a heuristic cost measurement method is developed for U-line balancing. The key idea in the core of this research is a task's marginal desirability for assignment at a given station. This idea is embedded in a beam search heuristic for solving both the straight line and U-line balancing problem. Extensive computational experiments and simulation experiments are made with well-known problems in the literature under the assumption of normally distributed task processing times. The quality of the solutions found by beam search for the straight-line balancing problem is compared to an existing method in literature. A simulation model of the assembly design is constructed and sample results from the U-line balancing problem are tested against the simulation results. The algorithm presented in this thesis improves the objective function by up to 24 percent.

Keywords: Assembly Line Balancing, Stochastic Task Times, Beam Search,

ÖZET

RASSAL İŞ ZAMANLI DÜZ VE U TİPİ MONTAJ HATLARININ DENGELENMESİ

Halil Şekerci
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. İhsan Sabuncuoğlu
Ağustos 2003

Bu tezde Rassal İş Zamanlı Montaj Hatlarının Dengelenmesi problemi üzerinde çalışıldı. Araştırmamız hem iyi bilinen anuyumlu düz montaj hatlarını hem de anuyumlu U tipi montaj hatlarını istasyonlar arasında tampon envanterlerin yokluğunda incelemektedir. Amacımız işgücü maliyeti ve ürünü çevrimdışı montajlama maliyeti gibi iki bileşenli bir maliyet fonksiyonunu en azlamaktır. Maliyet düz hatlar için kesin, U tipi hatlar için ise sezgisel bir yöntemle hesaplanmaktadır. Bu araştırmanın temelinde yatan ana fikir bir işin verilen istasyondaki konuma atanması için marjinal istenilirliğinin belirlenmesidir. Bu fikir düz ve U tipi montaj hatlarının dengelenmesinde kullanılmak üzere bir ışın taraması sezgisel yönteminin içerisinde kullanılmıştır. İş zamanlarının normal dağılıma sahip olduğu varsayımı altında literatürdeki iyi bilinen problemler üzerinde kapsamlı hesapsal deneyler ve benzetim deneyleri gerçekleştirilmiştir. Işın taraması kullanılarak elde edilen sonuçların kalitesi düz montaj hatları için literatürdeki bir diğer yöntemin sonuçlarıyla karşılaştırılmıştır. Montaj hattının bir benzetim modeli kurularak U tipi montaj hatları için elde edilen sonuçlar benzetim modelinin sonuçlarıyla karşılaştırılmıştır. Bu tezde sunulan yöntem amaç fonksiyonunda % 24'lere varan iyileştirmeler sağlamıştır.

Anahtar Sözcükler: Montaj hattı dengeleme, Rassal iş zamanları, Işın taraması,

Anneme ve Babama,

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Prof. İhsan Sabuncuoğlu and Prof. Erdal Erel who supervised me through all stages of this research. They are certainly not only supervisors of this thesis, but also a shareholder of it.

I am also indebted to Asst. Prof. M. Murat Fadıloğlu for his accepting to read and review this thesis.

My special thanks goes to İlktuğ Çağatay Kepek, for his invaluable friendship through my whole university life. He made me, never feel alone and was an indispensable part of our intelligent talks.

I would also like to thank to my friends, Aykut Özsoy, Onur Özkök, Hüseyin Özsert, Sabri Çelik, Müge Yayla, Savaş Çevik, Gökhan Metan, Ali Koç, Evren Emek, Sibel Alumur, Ünal Akmeşe for their friendship and morale support all the time. I am indebted to many others whose names are not on this page but whose love is in my heart for sure.

Finally, I owe so much to my family. Without their support, I could have never been the man I am.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 A brief history

Development of assembly lines is perhaps one of the most important triumphs of the twentieth century. The advent of assembly line in production systems, triggered mass production and made many products available to the benefit of mankind at reasonable prices. Although the first assembly line is credited to Henry Ford who developed such a line in 1913 and used it to produce Ford automobiles, the analysis and analytical statement of the assembly line balancing problem dates back only to 1955 (Salveson 1955). Jackson (1956), Bowman (1960), Supnik and Solinger (1960), White and Hu (1961) later followed his work. Extensive research on the assembly line balancing problem (ALBP) has accumulated since then, but the structure of the problem consistently defied the development of exact algorithms. Several survey papers review the work published on the subject: Kilbridge and Wester (1962), Ingall (1965), Mastor (1970), Buxey *et al.* (1973), Johnson (1981), Baybars (1986b), Yano and Bolat (1989), Erel and Sarin (1998), Amen (2000). In this section the concept of assembly line and the problem of assembly line balancing is introduced on straight shaped and U-shaped line configurations.

1.2 Preliminaries of the ALBP

An assembly line is a production sequence of stations connected together by a material handling system, where parts are assembled together at stations to form an end product. In this system there are work elements to be performed each of which is called a task. A task is the smallest indivisible work element in the assembly process.

Several tasks are performed at a physical location by a single worker and other tasks are similarly performed by other workers at different stations. A station is a location along the line at which tasks are performed by completing the assembly operations.

Task performance time, t_i is the duration of task i, and cycle time C is the amount of time available at each station. Equivalently cycle time is defined as the amount of time elapsed between two successive units entering or leaving the assembly line. Accordingly, station time S_j is defined as the sum of task times of the tasks assigned to station j on the line.

After the line begins to give the first product, a partially assembled product remains at each station during each cycle, while the set of tasks assigned to this station is performed on it. The material handling system then moves all partially assembled parts forward to next station and a new cycle begins. Thus all the units at every station advance to their next station in sequence at the same time. This time point is the end of cycle time. Thus, if tasks are completed on a unit before the cycle time ends, the unit waits idle until the end of cycle time. Because of this synchronization in movement, these type of assembly lines are sometimes called as "synchronous lines".

Since there must exist at least one station and at least one task at each station, cycle time is bounded by the following relation :

$$\max_{i=1,..,N} t_i \le \max_{j=1,..,K} S_j \le C \le \sum_{i=1}^{N} t_i$$

Tasks are not completed arbitrarily, rather there exists a precedence relationship between the tasks, dictating the completion of some tasks before

others can be started. A precedence diagram depicts the ordering, in which tasks must be performed to achieve a successful assembly of the product. This precedence diagram is either represented by a network of tasks or by an upper triangular *NxN* matrix, where *N* is the number of tasks in the assembly process. In the network representation, an arc originating from task *i* and ending at task *j* represents that task *i* must be completed before task *j* can be begun. In the matrix representation the entry [i,j] is 1 if task *j* follows task *i* in the precedence diagram, otherwise it is zero. Network representation is illustrated in Figure 1.1.

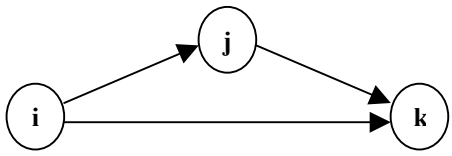


Figure 1.1: A precedence relationship diagram.

The assembly line balancing problem (ALBP) can be stated as assigning tasks to an ordered sequence of stations such that the precedence relations among the tasks are satisfied and some performance measure is optimized. The most commonly used objectives can be classified into two categories. In the first category one desires to minimize the number of stations given the cycle time. In this category we minimize number of stations subject to the following constraints.

(1) All tasks must be performed

(2) The work content in any station is less than or equal to the cycle time *C*.

(3) Precedence relations are not violated.

Notice that such an objective is equivalent to minimizing the total idle time, since

$$Idle\ time = K * C - \sum_{i=1}^{N} t_i$$

where *K* is the number of stations in the design, under consideration.

Thus, when idle time is minimized *K* is also minimized. The reduction is due to the fact that $\sum_{i=1}^{N} t_i$ is constant and *C* is given. This category is known as the Type I problem.

In the other category the objective is to minimize cycle time given the number of stations. This category is known as Type II problem. In both categories minimization is subject to precedence constraints.

A general integer programming formulation to the Type I problem is given as follows:

$$x_{ij} = \begin{cases} 1 & \text{if task } i \text{ is assigned to station } j \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in I \text{ and } \forall j \in J$$

Type I ALBP:

$$\text{Minimize} \quad \sum_{j=1}^{K} j x_{Nj} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{K} x_{ij} = 1 \qquad \text{for } i = 1,..,N \tag{2}$$

$$\sum_{j=1}^{K} j x_{ij} - \sum_{j=1}^{K} j x_{hj} \qquad \text{for } i = 1,..,N \text{ and } h \in F(i) \tag{3}$$

$$\sum_{i=1}^{N} t_i x_{ij} \le C \qquad \text{for } j = 1,..,K \tag{4}$$

$$x_{ij} = \{0,1\} \qquad \text{for } i = 1,..,N;\ j = 1,..,K \tag{5}$$

N represents the number of tasks in the problem, K represents the total number of stations in the design ($K \le N$), and $F(i)$ represents the set of tasks that are immediate precedence followers of task i.

In this formulation the objective function (1) minimizes the number of stations opened by minimizing the station number assigned to the terminating task. Constraint (2) is known as assignment constraint and states that each task is assigned exactly to one station. Constraint (3) is the precedence constraint and states that all predecessors of task i must previously be assigned in order to assign it to a station. Constraint (4) is the cycle time constraint and states that station times can't exceed the cycle time. Constraint (5) is the nondivisibility constraint of tasks.

Although the problem is easy to formulate, it has enormously large number of feasible solutions. Ignoring the precedence constraints, there are $N!$ different orderings possible. Precedence relations decrease the number of feasible solutions

drastically, but nevertheless the solution space is still too large to enumerate. Both Type I and Type II assembly line balancing problems are known to be *NP-hard* because the partition problem is known to be *NP-hard* (Papadimitriou, 1982). There is a vast number of heuristics and exact procedures in the literature to solve this problem.

There are also some other objectives offered in the literature other than the ones mentioned above. *Smoothness index* is a measure of how uniformly the workload is distributed among stations and is given by

$$S.I. = \sqrt{\sum_{j=1}^{K} (s_{\max} - s_j)^2}$$

Here s_j represents the total mean task duration at station j. S_{max} is the maximum of these statistics among all stations.

A measure of efficiency is balance delay, which is the ratio of the total idle time and the total time spent by a product moving from beginning to the end of line. It is given by:

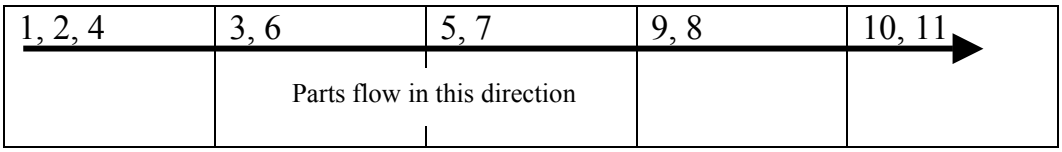
$$B.D = \left(\frac{100 * (KC - \sum_{i=1}^{N} t_i)}{KC} \right)$$

Balance delay measures the idle percent of time that the unit spends on line.

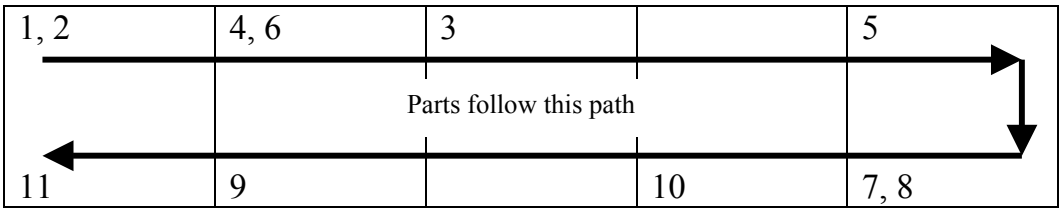
1.3 Preliminaries of U-line balancing

The key difference between the traditional (straight) assembly line balancing problem and the U-line balancing problem is the following: In the straight line balancing problem, units to be processed enter the line from the head of the line and proceed their way to next station as operations are completed on them. Finally completed units leave the line from the end of the line. Hence the units flow in one direction which is from the head of the line to its rear. However, in a U-line units enter the line and traverse all stations from first to last and return all the way back from last station to first. In U-line configuration a station has at any time two units, one moving in forward direction and the other in backward direction. Workers at any station first complete the necessary tasks on the forward moving

unit, then they may turn to finish tasks on backward moving unit. Hence the completed parts leave the line from the first station. Straight line and U-line configurations are illustrated in Figure 1.2.



A straight line configuration with 11 tasks assigned to stations.



A U-line configuration with 11 tasks assigned to stations.

Figure 1.2: Straight and U-line configurations.

Feasible U-line designs can be generated quite easily; the procedure is very similar to the straight line balancing case but proceeds in both forward and backward directions. In the straight line balancing problem, tasks are selected from a set of available tasks for assignment in order to form a station. These tasks are the ones whose predecessors have already been assigned. In the U-line balancing problem, the set of tasks available for assignment is the union of the set of tasks whose predecessors and successors have already been assigned. In other words, tasks with all predecessors assigned, are available for assignment in forward direction. Similarly tasks with all successors assigned, are available for assignment in backward direction.

The need for U-line configuration in manufacturing environments arises from attempts to improve productivity and increase flexibility. Miltenburg and Wijngaard (1994) state the following advantages of U-line configurations:

1. Quick response to changes in environment (machine breakdowns, worker absenteeism etc.)
2. Ease to adapt to changes in cycle time because of high potential to rebalance the line.
3. High level of participation between workers.
4. Flexibility for adding or removing workers
5. Require at most the same or fewer amount of stations than traditional lines.

U-lines also present some operational difficulties such as scheduling the movement of workers, dispatching jobs, etc. Moreover the line balancing problem of a U-line is much more complicated than the traditional line due to increased search space.

An integer programming formulation of U-line balancing problem due to Urban (1998) is as follows:

$$Let$$

$$m_{\min} = theoretical\ minimum\ number\ of\ stations$$

$$\lceil m_{\min} \rceil \le m^* \le m_{\max} \le n$$

$$x_{ij} = \begin{cases} 1\ if\ task\ i\ of\ the\ original\ network\ is\ assigned\ to\ station\ j \\ 0\ otherwise \end{cases}$$

$$y_{ij} = \begin{cases} 1\ if\ task\ i\ of\ the\ phantom\ network\ is\ assigned\ to\ station\ j \\ 0\ otherwise \end{cases}$$

$$z_j = \begin{cases} 1\ if\ station\ j\ is\ utilized; i.e., it\ is\ assigned\ tasks \\ 0\ otherwise \end{cases}$$

$$Minimize \sum_{j=\lceil m_{\min} \rceil+1}^{m_{\max}} z_j$$

$$subject\ to:$$

$$\sum_{j=1}^{m_{\max}} \left(x_{ij} + y_{ij} \right) = 1 \quad for\ i = 1,..,n \qquad (1)$$

$$\sum_{i=1}^{n} t_i \left(x_{ij} + y_{ij} \right) \le C \quad for\ j = 1,..,\lceil m_{\min} \rceil \qquad (2)$$

$$\sum_{i=1}^{n} t_i \left(x_{ij} + y_{ij} \right) \le C z_j \quad for\ j = \lceil m_{\min} \rceil + 1,..,m_{\max} \qquad (3)$$

$$\sum_{j=1}^{m_{\max}} \left(m_{\max} - j + 1 \right) \left(x_{rj} - x_{sj} \right) \ge 0 \quad for\ all\ (r,s) \in P, \qquad (4)$$

$$\sum_{j=1}^{m_{\max}} \left(m_{\max} - j + 1 \right) \left(y_{sj} - y_{rj} \right) \ge 0 \quad for\ all\ (r,s) \in P, \qquad (5)$$

$$x_{ij}, y_{ij}, z_j \in \{0,1\} \quad for\ all\ i, j.$$

When making assignments to stations two copies of the precedence network is used. One copy is considered for forward assignments and the other is used for backward assignments. The copy of the precedence network is also called as "phantom network". Forward assignments are made through the original network and backward assignments are made through the phantom network . In this formulation, *P* is the precedence set for which the element *(r,s)* indicates that task *r* immediately precedes task *s*.

Constraint (1) ensures that every task is assigned to only one station either in original or phantom network. Constraint (2) and (3) ensures that sum of the task times assigned to each station does not exceed cycle time. Constraints (5) and (6) enforce the precedence relationships between tasks.

1.4 Variations of the balancing problem

The assembly line problem has not remained as originally formulated. In time there arose many varieties of the original problem such as mixed model line balancing, U-type line balancing, stochastic assembly line balancing, etc. One such category that deserves special attention is the one that assumes stochastic

task times rather than deterministic. In the stochastic assembly line balancing problem task times are assumed to be random variables. Thus with such a setting one cannot guarantee that all the tasks assigned at a station be completed within the cycle time. Therefore, stochastic line balancing problem considers incompletion to occur at stations and alternative policies to adapt in incompletion situations. Naturally, the objective function in the formulation of stochastic line balancing may be different from the ones in deterministic cases. Incompletion ratio (Suresh and Sahu 1994) and expected line operating cost (Kottas and Lau 1973, Silverman and Carter 1986) are two example objectives used in literature.

1.5 Importance of the problem

The motivation for this study stems from the fact that assembly lines play important roles in today's manufacturing technology and understanding their behavior under variability is crucial to a firm's competitiveness. In general, variability is known to be detrimental but at the same time impossible to eliminate totally. Production plants, although designed for perfect synchronization, unfortunately do not operate at full efficiency due to the considerable variability inherent to the system. Conway et al. (1987) mentions that even in today's manufacturing plants a value of 10 for the ratio of flow time to total processing time is hard to achieve. Since the laws governing the performance of manufacturing systems are not understood to the full extent, it would be useful to provide the manufacturer with some design principles and guidelines. Without doubt a generic line balancing heuristic that employs these principles will reveal valuable information to the manufacturer and this will in turn reduce the operating cost of the plant and increase its competitiveness.

The use of JIT production methods also initiated the need for multi-functional workers and proper design of machinery layout. This resulted in U-shaped production lines which improved visibility and communication between workers as well as reducing the number of stations. The number of stations required on a U-line is never more than that required on a traditional line. This property of U-lines indicates their importance on the cost of production. Although

U-lines are important, there is little amount of work available in the literature. Therefore we believe that this study will contribute to the U-line literature especially if we consider that the stochastic U-line balancing problem has only one published journal paper.

1.6 The scope of this study

The problem investigated in this thesis is single model stochastic paced assembly line with straight and U-type configurations. Since task times are considered stochastic, operating costs incurred by balancing the line are affected by the cost of manning the line (labor cost), and the cost that arises from not completing the tasks as the unit moves down the line. These two cost terms are inversely related because the line operates at a constant output rate and amount of work to complete each unit on the line remains constant. The more work assigned to a worker reduces the number of workers needed but however it also increases the probability that the allocated work will not be completed within the given cycle time. Thus, a balance is to be established between these two cost terms given the cycle time. Figure 1.3 illustrates the situation.

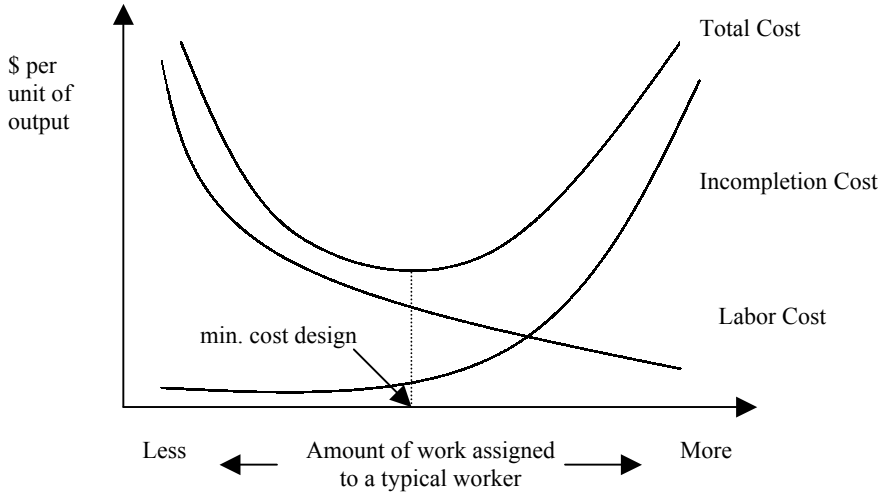


Figure 1.3: Single model stochastic line balancing costs. (From Kottas and Lau (1973))

Kottas and Lau (1973) report that industrial practice is to give some time allowance to the workers so that a variation in task time can be compensated without necessitating an incompletion. For this reason the industrial approach is to group the tasks into work assignments so that the sum of the expected task times does not exceed some specified percentage of the cycle time. However, as Kottas and Lau (1973) point out, this approach leaves two critical questions unanswered: Up to what percent of the cycle time should work stations be filled, and should this percentage be the same for all stations?

1.7. Assumptions of this study

The following assumptions are made about the assembly line considered in this research. These assumptions are the same as the ones adopted by Kottas and Lau (1973).

1. The cycle time and precedence relationships are the only restrictions on task assignments.
2. Each worker is paid the same wage regardless of the assignment
3. A task can only be begun if all its predecessors are completed.
4. The time to complete any task *i* is normally distributed with mean μ_i and standard deviation σ_i and further, the performance time of any task is independent of other task times and ordering of tasks within a station.
5. Whenever a task is not finished, the unit goes down the line with as many of the remaining tasks being completed as possible. All unfinished tasks are completed off-line. The cost to complete task *i* offline is not a function of what fraction of the task *i* was completed on the line.

The first three assumptions are very common in the assembly line balancing literature (Kottas and Lau (1973), Silverman and Carter (1986)). Normally distributed task times are widely used in the stochastic line balancing literature e.g. Mansoor (1968). Assumption 5 is just one of the possible line operating policies and closely approximates the situation often encountered in the assembly of automobiles and appliances (Kottas and Lau (1973)).

In this research stochastic straight line and stochastic U-type line balancing are studied. Therefore throughout the study tasks are assumed to come from a distribution function which is known in advance. The research concentrates on the ways to minimize the operating cost of these lines. To achieve this, line designs are generated and evaluated by heuristic methods.

1.8 Summary of work done

In this research a line balancing algorithm is developed for each of stochastic straight line balancing problem and stochastic U-line balancing problem. The objective in both problems is to minimize total cost of the line which comprises of labor cost and incompletion cost.

For the stochastic straight line balancing problem, Kottas and Lau's (1973) stochastic straight line balancing procedure and Kottas and Lau's (1976) straight line exact cost evaluation method is embedded in a beam search based algorithm to generate better designs in terms of cost than that of Kottas and Lau (1981). Several test problems are solved by the proposed method and the results are compared to that of Kottas and Lau's (1981) algorithm. Results indicate that the proposed heuristic can improve the solution found by Kottas and Lau's (1981) algorithm by up to 24 percent.

For the stochastic U-line balancing problem, Kottas and Lau's (1973) stochastic straight line balancing procedure is modified and streamlined. This procedure is embedded in a beam search based procedure together with a U-line cost evaluation heuristic which is developed by the author of this research. The efficiency of the heuristic cost estimation is compared with simulation results for several test problems. The results indicate for the test problems that the solution found by the heuristic is within 95% confidence interval of the simulation run results. The method developed for the U-line balancing problem is successful in that, it correctly estimates the cost of line with 95% confidence. Moreover, it is the first method in literature to estimate a cost based objective for the stochastic U-line balancing problem. Using the modified procedure together with beam search improves the quality of the solutions found.

1.9 Potential contributions to the literature

The contribution of this research to the existing literature is twofold. First the research presents a method for the stochastic U-type line balancing problem for which the first publication appeared only on February 2003. There is quite vast room for research on this field and U-type lines are becoming much more common as the JIT production philosophies get more popular. The second contribution is stemming from the heuristic method used in this research. Beam search, which is the main heuristic on which this research relies on, has never been used for single model assembly line balancing problem. In this respect this research is the first to use this search methodology for single model assembly line balancing problem. Beam search was previously being used in scheduling problems (Sabuncuoglu and Karabuk (1997), Sabuncuoglu and Bayiz (2000)) and for sequencing product types in mixed model assembly lines (Matanachai and Yano (2001)). Therefore this research presents a new usage area of beam search, namely the assembly line balancing. Moreover, the research gives valuable information on the basic principles of a good design over dominated designs.

Chapter 2

Literature Survey

ALBP's can be classified into four categories depending on whether task times are deterministic or stochastic and based on the variety of products assembled. These are: Single Model Deterministic (SMD), Single Model Stochastic (SMS), Multi/Mixed Model Deterministic (MMD) and Multi/Mixed Model Stochastic (MMS). The SMD version of the problem is the most common and the simplest version of the problem. In this version task times are known constants. The SMS version introduces stochastic task times, where the task times are not known in advance but rather the task time distribution is known. MMD version deals with the case when more than one type of item is produced on the same line and task times are known constants. Finally version MMS deals with producing more than one item on a single line where the task times are stochastic.

There are also other classification schemes based on movement of assembled parts along the line. In this scheme there are two distinct types, non-mechanical and moving belt lines. Operators on non-mechanical lines are unpaced since in these kind of lines a unit moves independent of other units when the process at the current station is complete. This type of transfer is called *asynchronous transfer*. Moving belt lines are simply characterized by a conveyor belt and are known as paced lines. In paced lines units at all stations move simultaneously. This type of transfer is called *synchronous transfer*.

In this survey single model deterministic and single model stochastic line balancing problems are covered. One classification of the ALBP and related solution procedures is presented in the survey paper of Erel and Sarin (1998). Their classification is introduced in Figure 2.1.

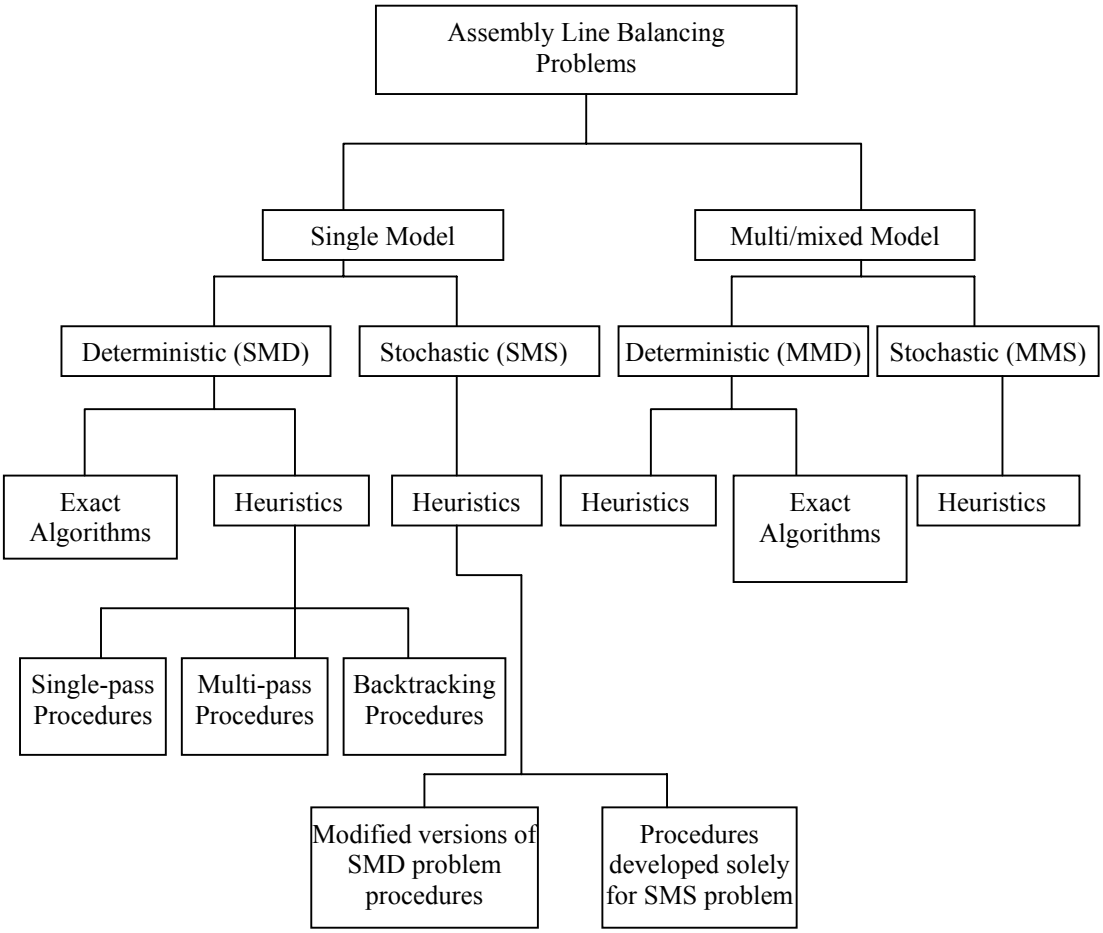


Figure 2.1: Classification of ALBP and related solution procedures

2.1 Single Model Deterministic ALBP

In this version of the problem, line is designed to produce only one model for which the task times are known with certainty. Thus, the problem is, given a finite set of tasks, a set of precedence constraints and a cycle time value, to assign tasks to an ordered sequence of stations such that the precedence relations are satisfied, total duration of tasks in any station does not exceed the cycle time and some performance measure is optimized. This problem is in the general class of sequencing and scheduling problems and is closely related to other problems in this class, such as single machine scheduling problem, bin packing and knapsack problem. Most of the studies on single model deterministic ALBP are about Type

I problem. There are two approaches in the literature for this problem: optimum-seeking algorithms and heuristics.

2.1.1 Optimum-Seeking Approaches

The exact methods in Type I and Type II problems can be treated under two main categories. In the first category the commonly used formulations are 0-1 IP and solution methodologies are enumerative techniques like branch-and-bound.

The first branch-and-bound algorithm was developed by Jackson (1956). Many other researchers followed his work with various optimum branching and optimum search strategies.

Johnson (1973) constructed a newest-node branch-and-bound algorithm and in 1981 he developed an improved version of his previous work (1973) by changing only the bounding mechanism.

Patterson and Albracht (1975) proposed a 0-1 IP formulation and the Fibonacci search procedure. In this method a sequence of 0-1 IP problems are examined to determine feasible solutions. They also used lower and upper bounds to reduce the number of variables.

Wee and Magazine (1981a) proposed a branch-and-bound algorithm that depends on two heuristics rather than the IP formulation. The first heuristic is called IUFFD (Immediate Update First-Fit Decreasing) and is a variation of the bin packing heuristic FFD (First-Fit Decreasing). The second heuristic is called IUBRPW (Immediate Update Backward Recursive Positional Weight) which is a reverse application of the well-known RPW (Ranked Positional Weight) technique.

Talbot and Patterson (1984) constructed a general IP algorithm and used network cuts and chains in order to expedite the backtracking in the problem. They have obtained optimal solutions for assembly lines up to 100 tasks in reasonable computational time.

Johnson (1988) proposed a method called FABLE (Fast Algorithm for Balancing Lines Effectively) which is a depth-first branch-and-bound algorithm. He used eight fathoming rules to shorten the search time.

Hoffman (1992) developed a depth-first branch-and-bound algorithm called EUREKA. This procedure searches all the branches by considering the "theoretical minimum slack time" fathoming rule. The method starts with theoretical minimum number of stations and if the cumulative sum of station slack times exceeds "theoretical minimum total slack time" then all emanating branches are fathomed. This method requires much computational effort.

Nourie and Venta (1991) proposed a method called OptPack which is a depth first search algorithm that checks solutions in lexicographic order until optimum is found.

Klein and Scholl (1996) proposed a branch-and-bound algorithm named as SALOME-2 for the Type II problem adapted from SALOME-1 (Scholl and Klein, 1994). This method uses a new enumeration technique, local lower bound method together with unidirectional and bi-directional search mechanisms.

Sprecher (1999) offered a competitive branch-and-bound algorithm for the Type I problem. His algorithm relies on a precedence tree guided enumeration scheme. He reformulates the ALBP as a resource-constrained project scheduling problem with single renewable resource whose availability varies with time and then uses branch-and-bound to solve the problem.

Amen (2000) proposed an exact method for cost oriented assembly line problem. He introduces an exact backtracking method in which the enumeration process is limited by modified and new bounding rules.

In the second category are the algorithms based on DP. The very first algorithm in this category was developed by Jackson (1956) although it was not formulated using the conventional DP terminology. A few years later a new DP algorithm was reported by Held and Karp (1962). Schrage and Baker (1978) proposed an efficient method for generating feasible sets. In their method, they define the feasible subsets of tasks and enumerate all of them with a labeling scheme. Their work was followed by Kao and Queyranne (1982). They defined a minimum cost function with the minimum number of stations needed for all tasks in their procedure. Computational experience indicates that as the size of the problem grows, the computational effort involved in DP algorithms increase enormously.

2.1.2 Heuristic Solution Approaches

The problem size sometimes makes it almost impossible to solve optimally. Therefore heuristic solution methodologies are developed to save from computational time at the cost of not guaranteeing the optimal solution. Heuristic procedures in single model deterministic ALBP are classified in three categories.

In the first category a single-pass decision rule is used. Such procedures prioritize some task based on a single attribute of each task using a list processing scheme.

The first and well-known was constructed by Helgeson and Birnie (1961) under the name Ranked Positional Weight Technique (RPWT). In this technique each task is given a weight equal to sum of its task time and task times of its followers. Then tasks are listed in decreasing weight and selection is made in that order as long as the cycle time and precedence constraints are not violated. If precedence constraints or cycle time constraint is violated, next task in the list is considered. If no further task can be assigned to the station, a new station is opened. Though its popularity, the method is shown to give very poor solutions by Ignall (1965) and by Mastor (1970) in their example problems.

There are other similar procedures which rank tasks according to some rule and selecting the highest rank task. Kilbridge and Wester (1961), proposed another heuristic that groups the tasks into columns in the precedence diagram and assigns them to stations by shifting their place in between groups.

Baybars (1986b) developed a heuristic that combines some tasks to reduce the size of the problem. Then he decomposes problem into smaller sub-problems to seek their solutions and finally he combines these solutions and decomposes tasks to reach the solution of the problem.

Wee and Magazine (1982) developed two heuristic procedures named RA (Rank-and-Assign) and GFF (Generalized First-Fit). These heuristics assign numerical scores to all tasks, ranks them in the descending order and selects them according to their rank and precedence relations.

The second category belongs to multiple-pass procedures. Arcus's (1966) technique called 'Computer Method of Sequencing Operations for Assembly Lines' (COMSOAL) is well-known example in this category. The main idea in COMSOAL is random generation of a feasible sequence. The method determines the available tasks for assignment at every iteration and selects randomly among the available tasks to fill the remaining station time. The author also used variations of the method by biasing the selection of tasks available for assignment. Among the variants the combined method gave the best results.

Later, Schofield (1979), Nksau and Leung (1995) constructed similar procedures in which best design is selected among several generated.

Hackman, Magazine and Wee (1989) developed several heuristic fathoming rules for the branch-and-bound algorithm so that the size of the problem is reduced.

The third and the last category comprises procedures that try to improve a solution or a station assignment by some iterative backtracking methods. An example to this category is the two phase procedure of Moodie and Young (1965) where in the first phase a preliminary balance is obtained by selecting among the available tasks the one with the largest performance times. In the second phase of this algorithm tasks are transferred between stations so that idle time is evenly distributed among the stations. Chiang (1998) uses tabu search for the ALBP. In his paper he considers four different approaches that use either first or best improvement strategies with or without task aggregation. Goncalves and Almeida (2002) used a hybrid genetic algorithm for ALBP. Their chromosome representation of the problem is based on random keys. The assignment of tasks to stations are made by some heuristic rules. They also use a local search to improve the solution.

2.2 Single Model Stochastic ALBP

The Stochastic Assembly Line Balancing Problem can be stated as assigning a set of tasks to an ordered sequence of stations, where performance times of tasks are distributed according to a probability distribution, subject to precedence constraints such that some performance measure is optimized. Now that the task times are random variables, a task can be incomplete either because the task is not completed within cycle time C or it is the precedence follower of another incomplete task. Incompletions reduce the efficiency of the line because they decrease throughput. So an incompletion cost term is associated with SALBP and this term depends on how incompletions are handled. Incompletions can be completed off the line and in this case the cost includes the labor cost of completing the task off the line. Incompletions can be handled by other ways as follows:

(1) The entire line can be stopped for the time necessary to complete the incomplete task

(2) Incomplete products can be inspected and repaired at special stations strategically located along the line.

(3) A skilled team can serve as a mobile repair station to help where needed.

When the task performance times are assumed to be random, the station time may exceed the cycle time C. As a result the enumeration and evaluation of the feasible solutions is much complex in the stochastic case. Hence the effort in the SALBP is limited to heuristics.

The stochasticity of task times are recognized to be normally distributed by several authors (Moodie and Young 1965, Mansoor and Ben-Tuvia 1966, Kottas and Lau 1973, Silverman and Carter 1986, Yano and Bolat 1989), however there are exceptions to this (Arcus 1966, Raouf and Tsui 1982).

The solution procedures to the SALBP can be classified into three categories. The first category involves modified versions of the solution procedures for SMD. The formulations in this category attempts to minimize labor cost either by filling the station up to a predetermined portion of the cycle time ($S_j \leq aC$, for all j, where $0<a\leq 1$) or they try to minimize labor cost by providing that at each station

there is at least a given probability of completing the work within the cycle time *C*. The two formulations are shown to be equivalent under certain circumstances.

Moodie and Young (1965), in order to provide an allowance at each station, computes the station time as:

$$s_j = \sum_{i \in K_j} \mu_i + r \sqrt{\sum_{i \in K_j} \sigma_i^2}$$

where *r* is a constant multiplier. Assuming that the tasks are independent a confidence level can be determined by adjusting *r*. Typical values for this parameter takes values around 1 (Moodie and Young (1965)).

Kao (1976) used a DP procedure to minimize the number of stations, while satisfying the precedence constraints and the constraint that, for all j, $P(S_j \leq C) \geq \alpha$, where α is the given lower bound.

Suresh and Sahu (1994) used simulated annealing to solve the SMS problem with the objectives of minimizing the smoothness index and the probability of line stoppage.

In the second category simulation is used to examine the problem and compare its deterministic and stochastic versions. Reeve and Thomas (1973) used a procedure that starts with an initial balance and rearranges tasks such that the probability of one or more tasks exceeding the cycle time is minimized. Buxey *et al.* (1973) used Monte Carlo simulation to examine SMS assembly lines. Driscoll and Abdel-Shafi (1985) used a balancing procedure similar to ranked positional weight technique and linked it with simulation to evaluate the performance of solutions.

The third category involves procedures developed solely for the SALBP. Kottas and Lau (1973) developed a heuristic procedure which attempts to minimize the total cost function comprised of total labor cost and total expected incompletion cost. They assume that whenever a task is not finished, the unit goes down the line with as many of the remaining tasks being completed as possible. In their procedure a task is assigned to the current station only if its anticipated labor savings are greater than its expected incompletion cost. Kottas and au (1981) developed an extension of their earlier work in which they use several selection

rules to generate several promising line designs. Sarin *et al*. (1997) developed an enumeration based approximation methodology. The proposed procedure divides the problem into sub-problems and obtains an initial solution to each sub-problem by a DP procedure. These solutions are then improved by using a branch and bound type of procedure. Finally these improved solutions are appended to each other to obtain the final solution. Erel *et al.* (1999) developed a methodology for SALBP. In their method they get an initial solution using dynamic programming and try to improve this solution by using a branch-and-bound procedure which uses approximate solution instead of lower bounds for fathoming nodes. A summary of the work done in literature is given in Table 2.1.

2.3 Single model deterministic ULBP

The literature on U-lines is sparse and new as compared to the traditional straight lines. In this literature there are two distinct groups. One concentrates on identifying the important design factors an their effects on the performance of U-lines. The other group, which is core to the scope of this research, concentrates on the problem of balancing U-type assembly systems to minimize either cycle time or the number of stations. Monden (1993) brought the U-lines to the attention of scientific community and since then the literature on U- lines accumulated at an increasing rate.

Miltenburg (2001-1) studied the effect of breakdowns on synchronized U-shaped production lines. This study assumes that small buffer inventories are placed between stations to reduce the effect of breakdowns. Miltenburg used line effectiveness, which is percent of time the line is up, as performance measure with constant repair rate and failure rate is assumed to be a linear function of the work done in the station. He suggested a markov chain model to investigate the effect of breakdowns and showed that U-lines dominate straight lines in terms of effectiveness in the presence of buffer inventories.

Nakade and Ohno (1999) worked on the optimal worker allocation problem on U-shaped production lines. They assume no buffer inventory between stations and derive a lower bound for the number of workers under the required

cycle time and propose an algorithm to find the allocation of workers to the line that minimizes the cycle time under the minimum number of workers which satisfies the demand.

Miltenburg (2001-2) published his tutorial-like study which analyzes one-piece flow manufacturing on U-shaped production lines. In his study he gave valuable ideas on designing the U-shaped lines, determining when one-piece flow manufacturing is appropriate etc. He also gave an integer programming, a dynamic programming and a Markov Chain representation of the problem and studied several example problems.

Miltenburg (2001-3) studied the U-shaped production lines. He described several U-line layouts such as simple U-lines, embedded U-lines, multi-lines in a single U, doubly dependent U-lines etc. He also gave examples of experiences of manufacturing companies with U-lines and the use of them in JIT production systems.

Erel, Sabuncuoglu and Aksu (2001) used simulated annealing to balance U-lines. In their study they try to minimize the number of stations. To achieve this they started with an initial solution and tried to decrease the number of stations by relaxing the cycle time constraint. Interestingly, they used simulated annealing to restore feasibility and used swapping and inserting as the neighborhood generation strategy. The authors also experimented the algorithm on several problems of varying size.

Scholl and Klein (1999) developed an algorithm for the several types of the U-line assembly line balancing problem. These types are UALBP-1 in which the number of stations are minimized given the cycle time, UALBP-2 in which cycle time is minimized given the number of stations and UALBP-E in which line efficiency is maximized with cycle time and number of stations are free to take any value. In their study line efficiency is defined as the sum of task times over number of stations multiplied with cycle time. Their solution methodology is branch and bound which uses several dominance rules and branching strategies. The authors show that their results are optimal for small size problems for which the optimal solutions are known.

Table 2.1: Summary of literature on stochastic paced straight line balancing problem

	AUTHOR	OBJECTIVE FUNCTION	SIZE OF PROBLEM INSTANCES	COMPARISON WITH OTHERS	CPU TIME	SOLUTION METHODOLOGY
1	Erel et al. (1999)	Incompletion cost + labor cost	11 to 60 tasks	Kottas and Lau	Up to 380 seconds	Branch and bound like heuristic
2	Gokcen H. (1999)	Incompletion cost + labor cost	11 task problem	Kottas and Lau	-	
3	Suresh and Sahu (1994)	Incompletion probability	9 to 48 tasks	Moodie, Reeve	101 sec for 21 task prob.	Simulated Annealing
4	Carraway R. (1989)	Number of stations				Dynamic Programming
5	Raouf and Tsui (1982)	Variation within a station	11 to 21 tasks	Moodie and Young	-	Priority based single pass heuristic
6	Kottas and Lau (1981)	Incompletion cost + labor cost		Moodie and Young		Single pass heuristic
7	Kao (1976)	Number of stations		-		
8	Kottas and Lau (1976)	Incompletion cost + labor cost		-	Up to 300 seconds	Enumeration based exact method
9	Kottas and Lau (1973)	Incompletion cost + labor cost	11 task problem	-	-	Single pass heuristic
10	Reeve and Thomas (1973)	Incompletion probability	21 to 70 tasks	-	Up to 63 seconds	Branch and bound like heuristic
11	Moodie and Young (1965)	Smoothness Index	21 to 70 tasks	IBM-Westinghouse	Up to 90 sec.	Single pass heuristic

Miltenburg and Wijngaard (1994) describe the U-line balancing problem and the importance of such lines in JIT production systems. They also show that classical assembly line balancing algorithms can be streamlined to solve U-line balancing problem. They illustrate this idea by a dynamic programming procedure and a heuristic ranked positional weight method. The authors also report some good computational results on some problems with number of tasks ranging from 7 to 111.

Miltenburg (1997) suggested a dynamic programming algorithm to minimize the number of stations subject to precedence, cycle time and location constraints. The author also adopts a secondary objective to concentrate the idle time in one station so that improvement efforts can be focused in that station in accordance with the JIT principles. The author's method however, does not prove to be effective for problems with size more than 22 and with sparse precedence graphs.

Urban (1998) presented an integer programming formulation for the U-lines. He solved the Type-I ULBP in which number of stations is minimized subject to cycle time and precedence relations. He solved problems up to 45 tasks with this formulation with the computation time ranging from 1 second to two hours.

2.4 Single model stochastic U-Line

Guerriero and Miltenburg (2003) suggested an exact recursive algorithm for the U-line balancing problem in which the objective is lexicographic minimization first over the number of stations and then over the incompletion probability in the last station. This paper assumes that task times have any distribution function and hence differ from any other work in the literature which assumes deterministic task times. The authors also make computational experiment for problems of size up to 30 tasks. The computation time however, exceeds 30 minutes in some instances. A summary of work done in literature is given in Table 2.2.

Table 2.2: Summary of work done on paced U-line balancing problem

	AUTHOR	OBJECTIVE FUNCTION	SIZE OF PROBLEM INSTANCES	COMPARISON WITH OTHERS	CPU TIME	SOLUTION METHODOLOGY
1	Guerriero and Miltenburg (2003)	Number of stations	7 to 35 tasks	-	May exceed 30 min	Recursive enumeration
2	Erel et al. (2001)	Number of stations	Over 45 task problems	Scholl and Klein	Up to 96 sec	Simulated Annealing
3	Scholl and Klein (1999)	Number of stations, cycle time and both	Up to 297 tasks	Known optimal solutions	Up to 89 sec	Branch and bound based heuristic
4	Sparling (1998)	Number of stations	Up to 18 tasks	-	-	Heuristic
5	Miltenburg (1998)	Number of stations	Up to 22 tasks	-	Up to 23 minutes	DP-based exact algorithm
6	Urban (1998)	Number of stations	Up to 45 tasks	Miltenburg	Up to 20 minutes	IP formulation
7	Miltenburg and Sparling (1995)	Number of stations	Up to 40 tasks	-	-	DP and depth-first branch and bound
8	Miltenburg and Sparling (1994)	Number of stations	Up to 11 for DP 111 for RPWT	RPWT	-	DP and RPWT

Chapter 3

Proposed Method

3.1 Structure of beam search

The method proposed to solve the problems stated in chapter 1 is a heuristic based on beam search. Beam search is a fast and approximate branch-and-bound method which operates on a search tree. However it differs from branch-and-bound because a certain number of best paths is selected and the rest is permanently pruned. Thus, at any level in the search tree only the promising nodes are kept for further branching and the other nodes are simply ignored. Beam search moves downward from the best β promising nodes at each level and β is called the beam width. Hence the heuristic is a partial enumeration technique which progresses level by level without backtracking.

In order to select the best β nodes, an estimate of the promise of each node is determined. This value can be determined in various ways: One way is to employ an evaluation function which estimates the minimum total costs of the best solution that can be obtained from the partial design represented by the node. In this case evaluation is based on the global view of the solution. Another approach would be to use one-step evaluation function which may rely on one or several surrogate measures. Unfortunately, there is a trade off between these approaches. One step evaluation is quick but may discard good solutions. On the other hand, a thorough evaluation by the global evaluation function is more accurate but computationally more expensive. Disregarding the complexity of global and local evaluation functions, beam search itself has polynomial time

complexity of $O(n^3)$ where n is the number of tasks in the problem (Sabuncuoglu and Karabuk 1998). The running time of the algorithm is polynomial in the size of the problem because a large part of the search tree is pruned off.

A filtering mechanism is also proposed in the literature to reduce the computational effort in beam search. With this mechanism a fast local evaluation function is used to discard some of the nodes based on their local evaluation function values, then only the remaining nodes are subjected to global evaluation. In this approach the number of nodes retained for global evaluation is called the *filter width* (α). However we would rather not use this approach because such a good local evaluation function is hard to find and the minimum cost objective is very sensitive to small changes in design. Hence it could be the case that a partial design that is locally evaluated to be bad can prove to be good when globally evaluated.

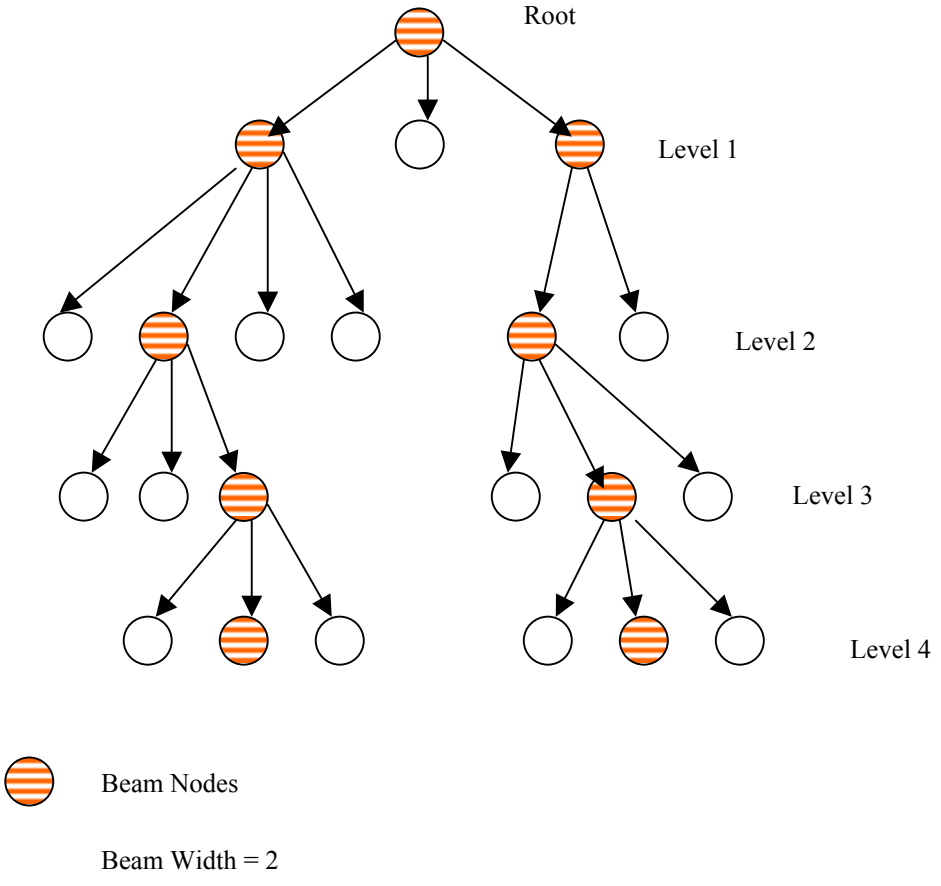


Figure 3.1: Representation of a beam search tree.

In Figure 3.1, a sample beam search tree is shown. We select the best β number of nodes from the nodes emanating from the root node by comparing the value of the global evaluation function of these nodes. After determining the first beam nodes at level 1, the algorithm is applied to these nodes independently and a partial tree is generated from each of them. Since there are beam width number of nodes in the current level and we progress by keeping one descendant only at each beam node, we have at any level beam width number of nodes and the search progresses from β parallel beams resulting in β different solutions in the end.

Without doubt, the quality of the solutions found by beam search depends both on the beam width β and quality of the global evaluation function.

This search technique was first used by Lowerre (1976) as an artificial intelligence method for the speech recognition problem. Later Fox (1983) used it to solve complex scheduling problems. In another study Chang *et al.* (1989) used beam search as a part of FMS scheduling algorithm. Sabuncuoglu and Karabuk (1997) developed a beam search based algorithm to evaluate scheduling approaches for flexible manufacturing systems. Leu *et al.* (1997) used beam search technique for sequencing mixed model assembly lines. Later Sabuncuoglu and Bayiz (1999) applied beam search for job shop scheduling.

Beam search has not been used in the context of assembly line balancing other than sequencing in mixed model lines. Thus, to the best of my knowledge this research is first to use beam search technique for balancing assembly lines. An overview of beam search and its applications can be found in Morton and Pentico (1993).

3.2 Beam search based algorithm for our problem

When using a beam search algorithm there are two important issues to consider: (1) search tree representation and (2) application of a search methodology.

3.2.1 Search tree representation

As previously mentioned, each node in the tree corresponds to a partial design. A partial design is an incomplete design where some tasks are allocated to opened stations but there are still tasks to assign and probably more stations to be opened. In this scheme a line between two nodes represents a decision to add a task to the existing station. Since at any level it may be desirable to close the station, we use a dummy task named *task 0*, the selection of which implies that the current station is closed and a new station is opened. Dummy task can be assigned again and again without being exhausted and is independent of any precedence relations. Once dummy task is assigned at a level, it is prohibited to assign it in the next level because this would mean opening and closing a station without assigning any task to it. Finally, the leaf nodes at the end of the tree correspond to complete designs. To facilitate understanding, the search tree representation of a small precedence diagram is given in Figure 3.2. According to the search tree in Figure 3.2, the path 1-2-0-3 implies that the tasks 1 and 2 are assigned to first station in the given order and task 3 is assigned to second station. The dashed nodes represent the final nodes of their paths.

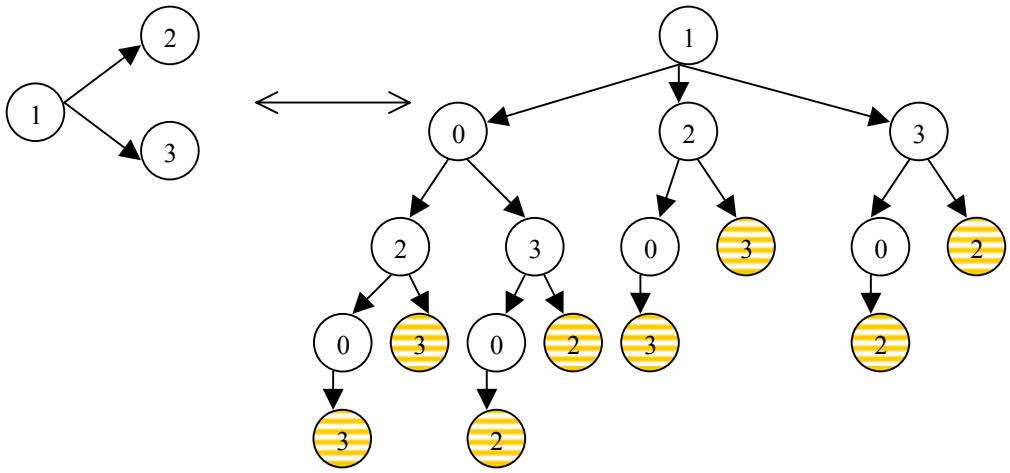


Figure 3.2: A precedence diagram and corresponding search tree

3.2.2 Search methodology

The second issue in beam search is determination of a search methodology. In the proposed algorithm beam search is used to perform search in the tree. No filtering mechanism is used, because filtering mechanism may cause to lose some good solutions for saving from computation time. However, our main concern is the quality of the solutions found rather than reduced computational time. Moreover there is no known filtering mechanism for assembly line balancing problem that is proved to work well. All the nodes at level 1 are globally evaluated to determine the best β number of promising nodes. The selected nodes become the first nodes of the β parallel beams. Subsequently the descendants of these selected nodes are globally evaluated to select the beam node at each parallel beam. If the number of nodes expanded at very first level are less than the specified beam width, then all the nodes in the following levels are expanded until the number of nodes at a level is greater than the specified beam width.

3.2.3 The search procedure

The procedural form of the proposed beam search algorithm is given as follows:

Step 0 :(*Initial node generation*) Determine the set of available tasks for assignment considering the precedence relations. These tasks constitute the level 1 nodes. Each node represents a partial design in which the selected task is assigned to the first position in the first station.

Step 1:(*checking the number of nodes*) If number of level 1 nodes is less than the specified beam width then expand nodes by generating further level nodes until the total number of nodes in the last level is greater than the specified beam width. Else go to Step 2.

Step 2: (*completing the partial designs*) Since these nodes represent only partially completed designs, they can't be evaluated globally. In order to evaluate these partially completed designs, they must be completed by assigning the remaining tasks by some heuristic rule. (More on this in the following sections)

Step 3: (*computing global evaluation function*) Compute the global evaluation function for all the nodes and select the best β of them. (initial beam nodes)

For each beam node:

Step 4: (*node generation*) Generate descendants of these beam nodes by considering the set of available tasks for that node. Set of available tasks for a node is determined by deducing all up to then assigned tasks for that node from the precedence diagram and choosing the ones with no unassigned predecessors for straight line balancing. For U-line balancing a task must have both its predecessors and followers assigned in order to be available for assignment. Consider also dummy task as the descendant provided that the beam node under consideration has not assigned it in the previous level. (closing the current station)

Step 4.1: (*computing global evaluation function*) Compute the global evaluation values of each of these nodes.

Step 4.2: (*selecting beam nodes*) For each beam in the tree select the node with the lowest global evaluation value(i.e., beam node). Go to step 2 and proceed until there is no task to assign.

Step 5 : (*selecting the solution design*) Among the beam width number of designs generated, select the one with the minimum objective value.

3.3 The evaluation mechanism for straight line

In the beam search based algorithm implemented in this research, when the search procedure comes to evaluate a node, which represents a partial design, it performs two operations. First the line design must be completed by some heuristic rule so that all tasks are assigned to their appropriate stations and secondly this complete design must be evaluated in terms of total expected cost

again by some other heuristic or exact procedure. Since this research does not propose any filter mechanisms all the generated nodes are evaluated globally.

3.3.1 Heuristic for completing partial designs

There are several variations of the main heuristic method considered for completing a partially complete design. These variations are all adapted from Kottas, Lau (1973), (1981).

The first heuristic considered is the same heuristic presented by Kottas and Lau (1973) with one slight modification. This heuristic successively builds up stations taking into account the precedence relationships. In order to achieve this, tasks that are available for assignment are classified in desirable list, which comprises tasks, whose placement in the current station increases cost no more than the cost of opening a new station. The selection is then made from sets by some priority rules depending on the status of the station under consideration. This heuristic proceeds in the following manner:

Given a node it first determines the set of assigned tasks and the precedence relations among the remaining tasks. To do this, precedence relationships are preserved for the unassigned tasks but the restrictions arising from tasks that are assigned are no more active. In other words the arcs originating from the tasks that are previously assigned are removed from the precedence diagram.

The very first step in this heuristic is determination of tasks available for assignment. Available tasks must have all their predecessors assigned. Hence if a task has no unassigned predecessor in the precedence diagram it is available for assignment. Such tasks are placed in a set called *available list*. The heuristic computes available tasks at each step to update the available list. If the available list happens to be empty, this means that there are no more tasks to assign and hence the line design is complete.

The next step in the heuristic is selection of tasks from the *available list* which are marginally desirable to perform. A task is considered marginally

desirable when its expected labor savings in the specific position under consideration is greater than or equal to its expected incompletion cost.

The labor savings is determined by how much the labor cost of performing task *k* at a new station will be reduced by performing it in the existing station. Since the marginal labor cost of performing a task at the existing station is zero, expected labor savings is equal to labor cost of the existing station.

The incompletion cost I_k , stemming from not completing task *k* is determined by both the cost of completing task *k* and the cost of completing all its precedence related followers off the line.

The expected incompletion cost for assigning task *k* in the existing station is equal to the cost I_k stemming from not completing task k in the line multiplied by probability P_k of not doing so within the cycle time *C*.

$P_k = 1\text{-}F(Z_k)$

where

$$Z_k = (C - \sum_{i \in S_k} \mu_i) / (\sum_{i \in S_k} \sigma_i^{\,2})^{1/2}$$

Marginally desirable tasks constitute the so called, *desirable list*. The heuristic computes the set of desirable tasks at each step. A station is closed when there are no desirable tasks available.

Among the tasks in desirable list some have virtual certainty of completion. These tasks comprise the *sure list* and is defined as those tasks for which $P_k < 0.005$.

There is one more occurrence to point out. It could be the case that some tasks may be available for assignment but not marginally desirable to assign. Such tasks comprise the so called *critical list*. Since any critical list task can't be made more desirable than by performing it first in the station they are assigned when the station is empty. A flow chart of the algorithm for completing the partial line design is given in Figure 3.3

The second heuristic considered is adapted from Kottas and Lau (1981) and in fact this heuristic is presented as an alternative to the existing one by Kottas and Lau (1973). This heuristic follows the same lines with the heuristic presented above. The only difference is the selection criteria of tasks for

assignment from desirable, critical or sure list. Contrary to the prior method this heuristic adapts random selection of tasks from these lists rather than the largest or lowest I_k criterion.

3.3.2 Procedure for evaluating designs

Once a line design is generated, its expected operating cost must be determined. For this reason we need an evaluation function. The global evaluation function used in beam search is adopted from the method developed by Kottas & Lau (1976). To be able to evaluate the incompletion cost associated with a paced line, it is necessary to be able to identify all the possible combinations of incomplete tasks, which occur in any unit coming off the line. This identification procedure utilizes the fact that each incompletion combination (*IC*) associated with a *K* station line design is uniquely represented by a *K*-tuple $(n_1,n_2,.....,n_K)$ where n_k is the number of tasks assigned to station *k* which are not completed due to lack of time.

A set generation process is used to identify the tasks belonging to each *IC* represented by the *K*-tuples as the latter are indexed through all their feasible values. Identifying the tasks comprising the *IC* represented by a given *K*-tuple is a matter of generating the appropriate sequence of sets. The process begins with station *1* and continues in *K* stages through station *K*. At each state *k*, the station *k* tasks belonging to the *IC* are identified. This is done by first deleting from station *k* any tasks which are precedence followers of incomplete tasks in station *1* through *k-1*. Remaining are the station k tasks which can be started. The last n_k startable tasks in this station belong to the *K*-tuple identified *IC* as do their precedence followers.

The probability P[*G*] of a given incompletion combination is calculated in the following manner (from Kottas and Lau (1976)):

$$P[G]=\prod_{k=1}^{K} P[G_k \mid H_k]$$

where G_k represents the tasks that are incomplete in station *k* due to lack of time, and H_k represents the set of tasks upstream of station *k* which are incomplete due to lack of time.

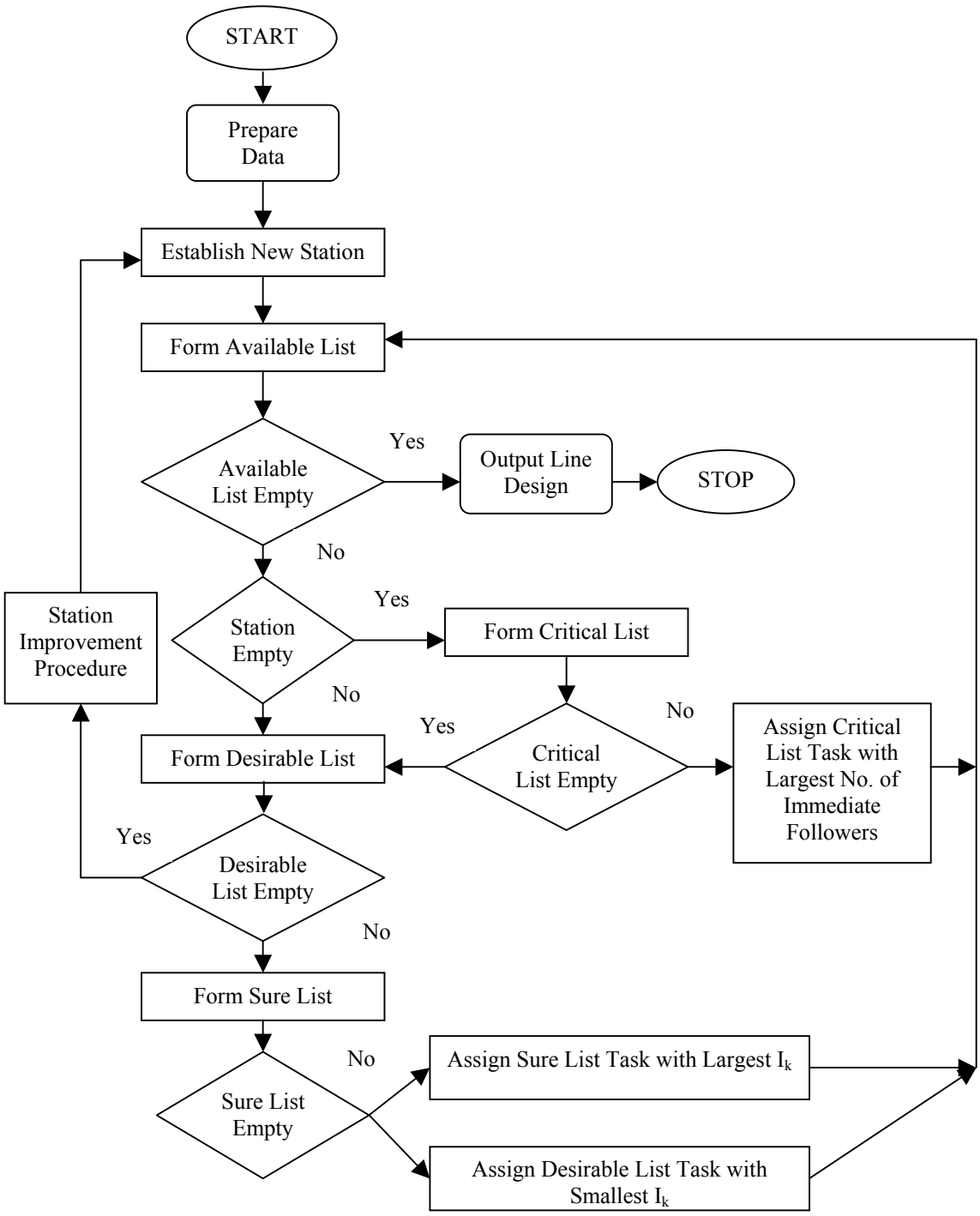


Figure 3.3: Flow chart of line balancing algorithm

There are two cases: when $G_k = \phi$ or $G_k \neq \phi$

$$P[G_k \mid H_k] = P[\sum_{i \in W_k} t_i \leq C] \; for \; k = 1,..K \;\; when \; G_k = \phi$$

when $G_k \neq \phi$

$$P[G_k \mid H_k] = P[(\sum_{i \in W_k} t_i \leq C) \cap (\sum_{i \in V_k} t_i > C)] \qquad \forall k$$

$$P[G_k \mid H_k] = P[(\sum_{i \in W_k} t_i \leq C)] P[(\sum_{i \in V_k} t_i > C) \mid \sum_{i \in W_k} t_i \leq C] \qquad \forall k$$

noting that

$$P[(\sum_{i \in V_k} t_i > C)] = P[(\sum_{i \in V_k} t_i > C) \mid (\sum_{i \in W_k} t_i \leq C)] P[(\sum_{i \in W_k} t_i \leq C)] + P[(\sum_{i \in V_k} t_i > C) \mid (\sum_{i \in W_k} t_i > C)] P[(\sum_{i \in W_k} t_i > C)]$$

we can write:

$$P[G_k \mid H_k] = P[(\sum_{i \in V_k} t_i > C)] - P[(\sum_{i \in W_k} t_i > C)] \qquad \forall k$$

$$P[G_k \mid H_k] = F[(C - \sum_{i \in W_k} \mu_i) / \sqrt{\sum_{i \in W_k} \sigma_i^2} - F[(C - \sum_{i \in V_k} \mu_i) / \sqrt{\sum_{i \in V_k} \sigma_i^2}$$

In this formulation W_k represents the set of tasks that are complete and V_k represents the set of tasks that are complete plus the first incomplete task.

Since objective function to be minimized is total unit labor cost plus the expected incompletion cost; the expected operating cost of a paced line is found in terms of time per unit by the following equation:

$$E_C = CK + \sum_{allG} I[G] P[G]$$

Below an example problem is given to detail how the cost evaluation procedure works. The example problem is taken from Kottas and Lau (1973).

Consider the example problem with the precedence diagram in Figure 3.4 and supporting data in Table 3.1:

A feasible design to this problem is :

	Line Design
Station 1	1,2,3,6
Station 2	4,5,8
Station 3	7,10,9,11

Table 3.2 summarizes all incompletion tuples and their respective occurrence probabilities.

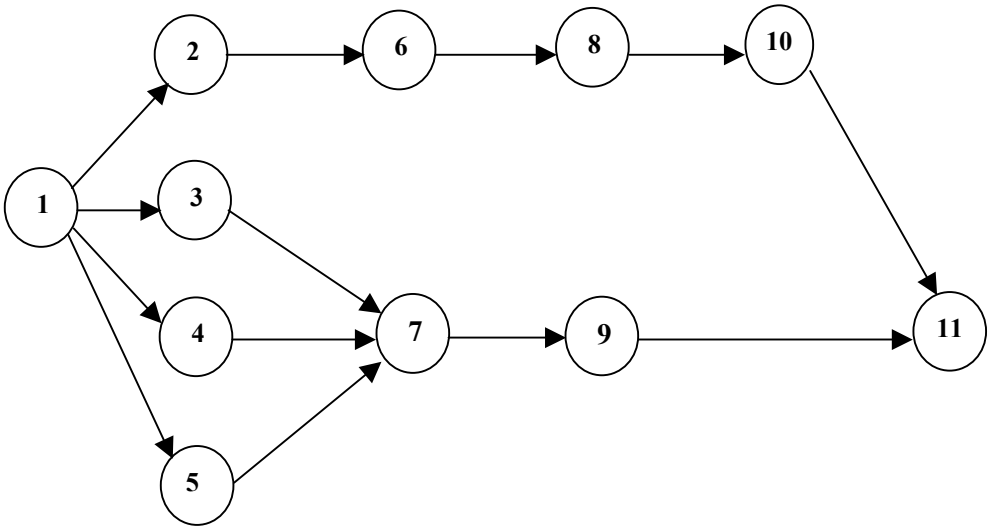


Figure 3.4: Precedence diagram of the example problem

Table 3.1: Supporting data for the example problem

Task No	Mean Task Time	Variance of Task Time	Task Incompletion Cost
1	4	0.8	5.6
2	2	0.4	2.8
3	8	1.6	11.2
4	8	1.6	11.2
5	3	0.6	4.2
6	1	0.2	1.4
7	3	0.6	4.2
8	3	0.6	4.2
9	1	0.2	1.4
10	8	1.6	11.2
11	4	0.8	5.6

Table 3.2: All incompletion combinations and their respective probabilities

IC's			Prob. IC. Occurs	
3-tuple representation	Time related incomplete tasks	Incompletion Cost	When C = 15	When C = 20
(0,0,1)	11	5.6	0.2485	0.0126
(0,0,2)	9	7.0	0.0083	0
(0,0,3)	9,10	18.2	0.00127	0
(0,0,4)	10,7	22.4	0	0
(0,1,0)	8	21.0	0.1358	0.000168
(0,1,1)	8,9	22.4	0	0
(0,1,2)	8,7	26.6	0	0
(0,2,0)	8,5	30.8	0.00175	0
(0,3,0)	8,5,4	42.0	0	0
(1,0,0)	6	22.4	0.2242	0.00178
(1,0,1)	6,9	23.8	0	0
(1,0,2)	6,7	28.0	0	0
(1,1,0)	6,5	32.3	0.000787	0
(1,2,0)	6,5,4	43.4	0	0
(2,0,0)	6,3	39.2	0.2741	0.000168
(2,1,0)	6,3,5	43.4	0.000963	0
(2,2,0)	6,3,5,4	54.6	0	0
(3,0,0)	3,2	42.0	0	0
(3,1,0)	3,2,5	46.2	0	0
(3,2,0)	3,2,5,4	57.4	0	0
(4,0,0)	1	63.0	0	0

At C = 15 min.	Total Exp. Inc. Cost =	20.2104
	Total Labor Cost =	45 = (15*3)
	Total Cost	65.2104
At C =20 min.	Total Exp. Inc. Cost =	0.1208
	Total Labor Cost =	60=(20*3)
	Total Cost	60.1208

3.4 The evaluation mechanism for U-line

Similar to the straight line balancing case we need a heuristic for completing partially generated designs as well as a heuristic for evaluating them.

3.4.1 Heuristic for completing partial designs

This heuristic follows the same lines with the heuristic for balancing straight lines with a slight modification in determining the set of available tasks (See Figure 3.2). As there are two units moving in different directions at a station in U-lines, in assigning tasks to stations one must consider not only the tasks whose predecessors have already been assigned, but also those tasks whose successors have already been assigned. In other words, tasks whose predecessors have already been assigned, are available for assignment in forward direction at a station whereas tasks whose successors have already been assigned, are also available for assignment in backward direction. Therefore the *available list* comprises not only tasks whose predecessors are assigned, but also comprises tasks whose followers are assigned.

The second and the last modification is about the task selection rule. Among the tasks that are marginally desirable for assignment, a subset is *sure list.* The tasks in this set have virtual certainty of completion. When a selection is to be made from the sure list, i.e. sure list is not empty, tasks available for forward assignment are considered according to "largest incompletion rule", whereas the tasks available for backward assignment are considered according to "lowest incompletion rule". This is reasonable because the worker at a station first performs the operations on forward direction and then turns back to perform operations on backward direction. If there are tasks available for both sides the selection is made in favor of forward assignment.

3.4.2 Heuristic for evaluating designs

Rather than the exact procedure for the straight line balancing case a simpler heuristic method is used for the U-line balancing problem. This heuristic assumes that, only the tasks that are allocated to later parts of task sequence in a station can be incomplete. In order to determine these tasks, the following method is adopted: First the station variance ξ^2_{sta} of station under consideration is calculated. Then a threshold value of $C\text{-}\alpha*\sqrt{\xi^2_{sta}}$ is calculated for each station. Here C is cycle time and α is a confidence parameter. In our study the number α is chosen as 2, which implies that the probability of a task being incomplete before this point in cycle time, is less than 0.03.

After determining the tasks that are candidate for incompletion, we need to estimate their share in incompletion cost. In order to achieve this we need to calculate their incompletion probability as well as total incompletion cost of their followers.

The total expected cost is approximated by the following relation:

$$Ex.Cost \approx \sum_{i \in candidates} \left(P\{First\ \ incompletion\ \ is\ \ on\ \ task\ \ i\} * r * (\mu_i + \sum_{j \in F(i)} \mu_j) \right)$$

where r is the off-line completion rate.

A task sequence is obtained by appending tasks assigned in backward direction, if any, to tasks assigned in forward direction. If task under consideration is in position j in the task sequence, then its incompletion probability within the station is calculated as:

$$P = \phi\left(\frac{C - \sum_{i=1}^{j-1} \mu_i}{\sqrt{\sum_{i=1}^{j-1} \sigma_i^2}} \right) - \phi\left(\frac{C - \sum_{i=1}^{j} \mu_i}{\sqrt{\sum_{i=1}^{j} \sigma_i^2}} \right)$$

where $\varphi(x)$ represents the area under standard normal distribution curve to the left of point x.

Further, this probability is multiplied by the probability that all other tasks that must be performed prior to the task under consideration is complete to get the probability that first incompletion is on task i.

Total incompletion cost in case of such an occurrence is bound below by :

$$r*(\mu_i + \sum_{j \in F(i)} \mu_j)$$

Hence, the heuristic tends to underestimate the cost of line. But the experimentation shows that results are quite reasonable.

An example problem is given below to illustrate the heuristic method:

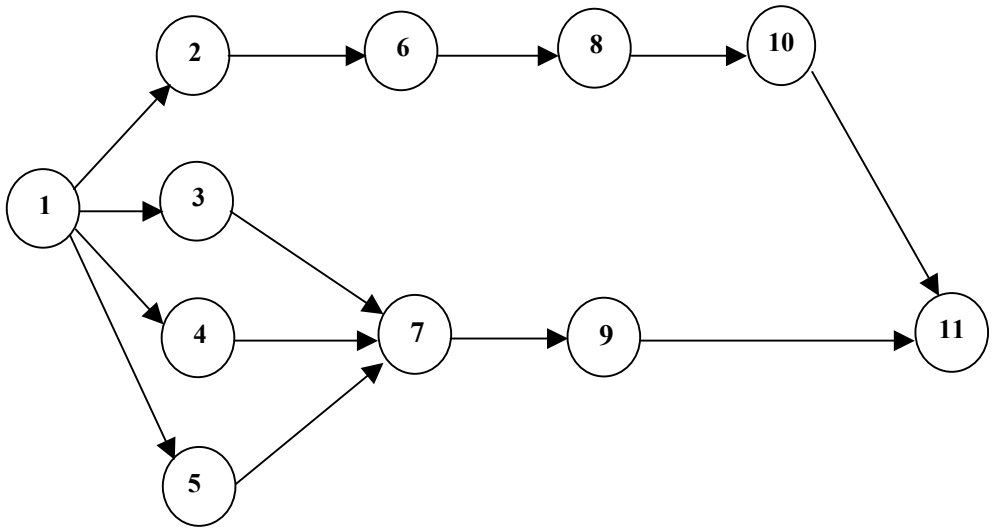


Figure 3.5: Precedence diagram of the example problem

Consider the following design:

1 11, 9	3, 4	2, 5, 6 10	7, 8

Table 3.3: Task means: (T_i *for the* i^{th} *task*)

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}
6	2	4	9	2	2	3	6	5	5	3

Table 3.4: Task variances: (σ^2_i *for the* i^{th} *task*)

σ^2_1	σ^2_2	σ^2_3	σ^2_4	σ^2_5	σ^2_6	σ^2_7	σ^2_8	σ^2_9	σ^2_{10}	σ^2_{11}
1.2	0.4	1	5	0.4	0.4	0.6	1.2	1	1	1.8

Let cycle time C = 15 and off-line completion rate r =1.5

Then, threshold mean times for stations are:

Station 1	Station 2	Station 3	Station 4
15-2*√(1.2+1+1.8) =11	15-2*√(1+5) =10.1	15-2*√(0.4+0.4+0.4+1) =12.03	15-2*√(0.6+1.2) =12.31

Thus, the critical tasks are 4 and 11.

P[4 is the first incomplete task]=P[Task 1 is complete]*P[Task 3 complete and 4 incomplete]

=Φ((15-6)/√1.2)* [Φ((15-4)/√1)- Φ((15-13)/√6)] = 0.209

P[11 is the first incomplete task] = P[Task 1 and 9 is complete, 11 incomplete] * P[no incompletion in station 2]* P[no incompletion in station 3]* P[no incompletion in station 4]

=[Φ((15-11)/√2.2)- Φ((15-14)/√4)]* Φ((15-13)/√6)* Φ((15-11)/√2.2)* Φ((15-9)/√1.8) = 0.2403

Hence the incompletion cost of the line is approximated as :

0.209*(1.5*(9+3+5+3))+0.2403*(1.5*(3))= 7.35

The procedural form of the heuristic is:

Begin

Input Design

Get threshold mean for all stations

Determine the critical tasks

For i = 1 to number of stations

For j = 1 to number of critical tasks at station i

Begin

Calculate probability that critical task *j* at station *i* is incomplete

Determine the tasks that can't be completed due to incompletion of critical task number *j*.

Calculate expected incompletion cost arising from incompletion of critical task number *j*.

End

End.

Chapter 4

Experimental Setting

In this section the parameters input to the described problem formulation are given and how they are determined is explained. Before experimentation with the proposed method is given, the literature is searched to understand the methods adopted by other researchers in this field. Below we give an overview of the approaches in literature as well as our approach under appropriate headings.

Cycle Time: Since the model proposed in this research minimizes the total expected cost given the cycle time, cycle time is an input to this research. Theoretically cycle time is a value bounded by maximum task duration and total task duration. $(\max_{i=1,..,N} t_i \le C \le \sum_{i=1}^{N} t_i)$ However, there is no universally agreed method to select cycle time. Miltenburg (2003) suggest selecting randomly from the first half of this range. This is reasonable because selecting cycle time value from the other half means that there will be few number of stations to the solution of the problem. This is undesirable because such a selection oversimplifies the problem. In this research we treat each problem under three different cycle time values and select the cycle time from the first half of the possible range trying to select as evenly as possible.

Task processing times: In this research task processing times are assumed to be stochastic. Hence a duration realization from a distribution is required for each task to be complete. In the literature there are several distributions used for stochastic process times. For example Magazine (2000) use shifted exponential distribution. The stochasticity of task times are recognized to be normally

distributed by several authors (Moodie and Young 1965, Mansoor and Ben-Tuvia 1966, Kottas and Lau 1973, Silverman and Carter 1986, Yano and Bolat 1989, Miltenburg 2003), however there are exceptions as well (Arcus 1966, Raouf and Tsui 1982). Since normally distributed processing times are most widely used in the literature, task processing times are assumed to come from normal distribution with known mean and variance. This research employs the standard test problems from literature in the experiments. Since these problems are generated for deterministic assembly line balancing, tasks have deterministic processing times. In order to use these test problems in our case, we need both a mean and a variance value for each task. This is achieved in the following manner. Task means are set to their deterministic processing times. In other words, the task processing times in the deterministic assembly line test problems are assumed to be the mean processing times in the stochastic case. The standard deviation term is obtained by multiplying the task means with a coefficient of variation. In the literature, coefficient of variation terms ranging from 0.1 to 0.4 are used with 0.1 indicating low variability and 0.4 high variability. In the past, Silverman and Carter (1986) used 0.1 as coefficient of variation for low variability and 0.25 for high variability. We use the same coefficient of variation terms in our experiments. These are 0.1 for low variability and 0.25 for high variability.

Off-line completion rates: As previously stated when a task is incomplete, it moves down the line with as many of the remaining tasks being completed provided that the precedence relations permit. However, when a unit is incomplete on the line its assumed to be completed off-line. The off-line cost for completing a task is assumed to be a multiple of the mean task processing time. In the literature there are several values used for this multiple value. Kottas and Lau (1976) use 1.4 as coefficient of variation. Silverman and Carter (1986) use three different levels: 1.5 for low off-line rate, 5 for medium off-line rate and 10 for high off-line rate. In this research we use 1.5 for low off-line rate and 5 for high off-line rate.

Test problems: As previously mentioned test problems are taken from the literature. Scholl and Klein (1999) collected these data sets and published at their

web site (http://www.bwl.tu-darmstadt.de/bwl3/). However, for easy reference the reader can access these data sets also from our web site:
(www.bilkent.edu.tr/~halils/albdata.zip).

Choice of beam width: As previously mentioned the choice of beam width is of crucial importance when evaluating the performance of a beam search heuristic because it is directly related with the size of the search tree enumerated and investigated. In the literature most application of beam search are reported to make no improvement in objective when beam width is raised above five. However, these results are most commonly for problems in scheduling applications. Since there is no previous work of beam search on ALBP, we have to make pilot runs on some test problem to get an initial idea of what beam width to use in ALBP. So the results of three test problems each taken at three different cycle time levels are presented in Figures 4.1 through 4.3.

The experimental results suggest that a good solution can be obtained when $\beta=2$ or $\beta=3$. Increasing the beam width further does not make any contribution to the quality of the solution found. It should be noted that a generalization can't be made by investigating these nine instances but nevertheless in this research higher beam width values will not be used for computational purposes.

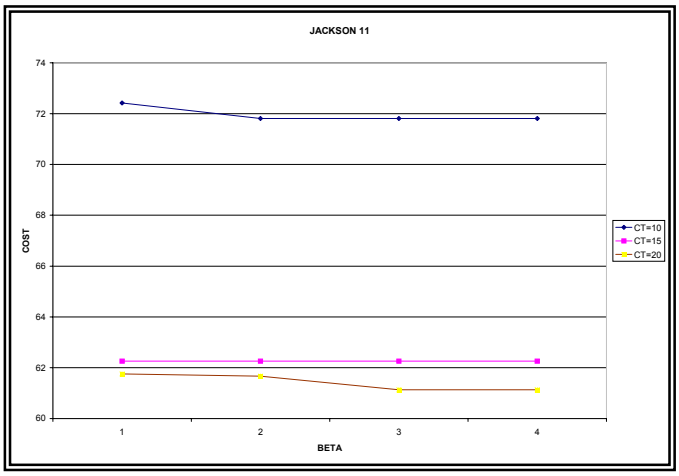


Figure 4.1: The impact of beam width for Jackson's 11 task problem.

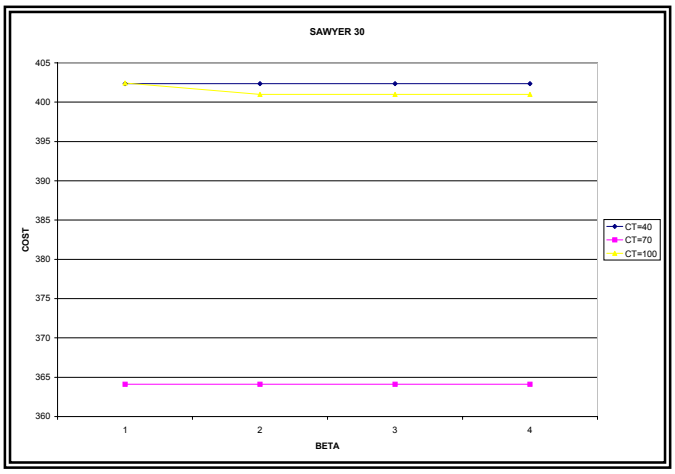


Figure 4.2: The impact of beam width for Sawyer's 30 task problem.

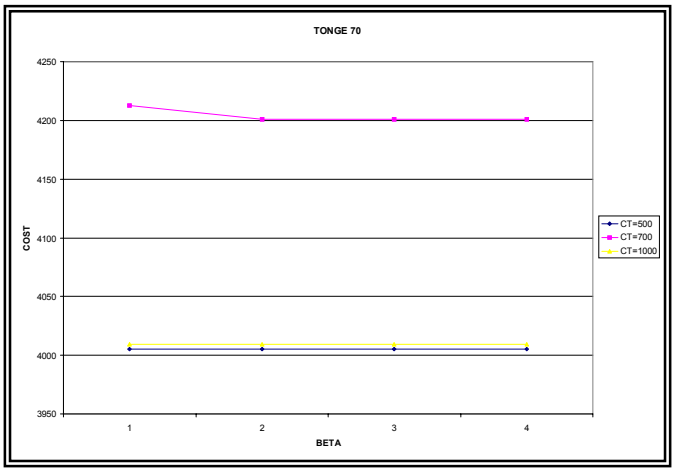


Figure 4.3: The impact of beam width for Tonge's 70 task problem.

Chapter 5

Computational Results

5.1 Computational Results for SLBP

In this section we present the results of the proposed beam search heuristic on several test problems studied. The test problems are well-known problems selected from literature and their size ranges from 11 tasks up to 70 tasks. These problems are solved under different cycle time, off-line completion rate and coefficient of variation settings. In order to compare our algorithm with the best compatible heuristic in literature we coded Kottas and Lau's (1981) algorithm. Kottas and Lau's (1981) algorithm is a single pass heuristic method that uses several selection rules for task assignments to stations. Briefly their method works in the following fashion:

1. Get the set of tasks available for assignment
2. From the set of tasks available for assignment, determine those that are marginally desirable to assign to the existing station
3. Select a task from the marginally desirable tasks by some selection rule. If the total mean task times at the existing station is less than a specified percent ($k\%$) of cycle time then use selection rules for the "early part" of cycle time. Else use selection rules for the "later part" of cycle time.
4. Evaluate the total cost of the line designs generated
5. Select the line design with the lowest cost

For "early part" of a station one of the two rules is used. These are:

1. Random selection
2. Select task with the largest task incompletion cost I_i .

For the "later part" of the station the one rule is selected from the following four rules:

1. Random selection
2. Select task with lowest task incompletion cost I_i .
3. Select task with highest mean task processing time (μ_i).
4. Select task with highest value of (μ_i/I_i).

Therefore, a total of 4*2=8 combinations of these selection rules can be used to generate a design. If the selection rule is a deterministic rule for both the "early part" and "later part" of the station the design generated is unique. However, whenever random selection rule is employed, different replications of the heuristic yields different designs. Thus the heuristic may be run again and again to get different solutions. In our study we used two levels for determining the "early part" and "later part" of a station. These levels are: *k=60%* and *k=80%*. We should also note that both algorithms are let to run for approximately same duration so that a fair comparison can be made.

The results found by the proposed heuristic are given in Tables 5.1 through 5.6. In this tables you can also find the results found by Kottas and Lau's (1981) heuristic and the percent improvement that the proposed heuristic makes over Kottas and Lau's (1981) algorithm. The algorithms are coded in Borland Pascal 7.0. and the results are obtained on a Pentium II 400 mhz processor PC.

Of the total of 72 instances solved, the beam search based heuristic beats the Kottas and Lau's (1981) heuristic in 64 instances with the percent improvement in objective function ranging between 0.1 % and 24 %. In 7 instances, Kottas and Lau's (1981) heuristic finds better results with improvement over the beam search based heuristic being no more than 0.67 %.

Since Kottas and Lau's (1981) algorithm is a single pass heuristic that runs very quickly, many random line designs can be generated and evaluated from scratch. Moreover, random selection rule makes it possible to explore a wide range of feasible good solutions. Most of the best designs found by Kottas and

Lau's algorithm are the ones generated by using random selection rule in either "early" or "later" parts of the station. However, there is one big disadvantage of Kottas and Lau's (1981) algorithm. This algorithm is not very structured because the basic motivation behind the algorithm is to generate many designs by incorporating randomness into selection. Thus at any iteration, selecting a task marginally desirable for assignment by some selection rule may make another task, which is currently desirable, undesirable at next iteration. This is because desirable set of tasks are updated after every task assignment to a station. Beam search based heuristic, on the other hand is more structured since it rather makes the task assignment after considering the assignment of all marginally desirable tasks one by one and then selecting the best. This one step look ahead property of beam search based heuristic makes it possible to keep the set of marginally desirable tasks nonempty as long as possible by selecting the "right task". Therefore beam search based heuristic can find solutions that involve fewer number of stations with higher incompletion costs. This could be very beneficial especially if the cycle time is high, because then closing a station brings a high reduction in cost function as compared to increase in incompletion cost. For example in Table 5.6 on line 11, reducing the number of stations from 4 to 3 saves from the cost by 1200 (since cycle time is 1200) at the expense of incurring additional 285 time units in off-line completion cost. This makes a 24 percent improvement in cost function which is very promising.

In searching for good designs, the objective in this thesis is to minimize a cost function made up of two parts: labor cost and incompletion cost. Generating line designs to minimize such an objective may favor some designs that produce incomplete units over designs that produce complete units at higher cost. However, this is not a big issue because such a situation can always be avoided by setting the off-line completion rate to a high value.

Table 5.1: Results for Jackson's 11 task problem

CT	Inc. Rate	C.V.	Beam Search Solution				Kottas & Lau Solution			Percent Improvement
			# of Stations	Labor Cost	Inc. Cost	CPU	# of Stations	Labor Cost	Inc. Cost	
10	1.5	0.15	6	60	1.96	4.3	7	70	0.629	13.99
		0.25	6	60	7.23	4.38	7	70	5.536	12.35
	5	0.15	6	60	5.34	3.84	7	70	2.094	10.34
		0.25	6	60	28.03	4.94	8	80	8.961	1.06
15	1.5	0.15	3	45	14.13	1.32	4	60	0.278	1.94
		0.25	4	60	2.75	2.1	4	60	2.827	0.12
	5	0.15	4	60	0.928	1.76	4	60	4.587	6.01
		0.25	4	60	10.87	2.32	5	75	1.575	8.05
20	1.5	0.15	3	60	0.7	1.1	3	60	0.85	0.25
		0.25	3	60	1.634	1.76	3	60	1.832	0.32
	5	0.15	3	60	0.23	0.98	3	60	0.23	0.00
		0.25	3	60	3.42	1.32	3	60	4.826	2.22

Table 5.2 Results for Mitchell's 21 task problem

CT	Inc. Rate	C.V.	Beam Search Solution				Kottas & Lau Solution			Percent Improvement
			# of Stations	Labor Cost	Inc. Cost	CPU	# of Stations	Labor Cost	Inc. Cost	
20	1,5	0,15	7	140	0,43	18,24	7	140	0,827	0,28
		0,25	7	140	6,88	14,06	7	140	8,438	1,06
	5	0,15	7	140	0,65	15,7	7	140	0,951	0,21
		0,25	8	160	5,79	15,6	8	160	10,386	2,77
30	1,5	0,15	4	120	2,57	7,24	4	120	3,774	0,98
		0,25	4	120	5,99	11,22	4	120	11,744	4,57
	5	0,15	5	150	0,18	11,98	5	150	5,582	3,60
		0,25	5	150	1,52	14,38	5	150	1,664	0,10
40	1,5	0,15	3	120	2,24	7,14	3	120	6,942	3,85
		0,25	3	120	3,14	9,66	3	120	3,797	0,53
	5	0,15	3	120	2,397	5,92	3	120	2,723	0,27
		0,25	4	160	0,97	7,46	4	160	0,38	-0,37

Table 5.3 Results for Sawyer's 30 task problem

CT	Inc. Rate	C.V.	Beam Search Solution				Kottas & Lau Solution			Percent Improvement
			# of Stations	Labor Cost	Inc. Cost	CPU	# of Stations	Labor Cost	Inc. Cost	
40	1.5	0.15	10	400	6.32	382.5	10	400	20.62	3.52
		0.25	11	440	27.13	461.6	11	440	31.849	1.01
	5	0.15	11	440	1.95	474.2	11	440	14.054	2.74
		0.25	12	480	20.53	631.8	12	480	28.704	1.63
70	1.5	0.15	5	350	21.06	91.8	5	350	23.946	0.78
		0.25	5	350	34.52	127.6	5	350	42.453	2.06
	5	0.15	6	420	3.15	100.1	6	420	2.673	-0.11
		0.25	6	420	11.73	160.9	6	420	14.587	0.66
100	1.5	0.15	4	400	4.28	50.6	4	400	3.312	-0.24
		0.25	4	400	4.84	38.7	4	400	6.161	0.33
	5	0.15	4	400	0.4	33.8	4	400	0.527	0.03
		0.25	4	400	7.89	61.64	4	400	8.676	0.19

Table 5.4: Results for Kilbrid's 45 task problem

CT	Inc. Rate	C.V.	Beam Search Solution				Kottas & Lau Solution			Percent Improvement
			# of Stations	Labor Cost	Inc. Cost	CPU	# of Stations	Labor Cost	Inc. Cost	
100	1.5	0.15	7	700	9.69	1763.05	7	700	12.596	0.41
		0.25	7	700	8.19	2255.13	7	700	13.938	0.81
	5	0.15	7	700	5.92	1812.15	7	700	7.074	0.16
		0.25	7	700	15.21	2353.1	7	700	19.819	0.64
150	1.5	0.15	4	600	8.21	454.35	4	600	9.95	0.29
		0.25	5	750	6.27	418.64	5	750	7.06	0.10
	5	0.15	5	750	5	473.73	5	750	3.948	-0.14
		0.25	5	750	6.824	412.71	5	750	10.976	0.55
200	1.5	0.15	3	600	3.42	126.66	3	600	4.309	0.15
		0.25	3	600	18.06	152.97	3	600	19.978	0.31
	5	0.15	3	600	9.6	119.3	3	600	11.533	0.32
		0.25	4	800	6.7	359.15	4	800	8.877	0.27

Table 5.5: Results for Warnecke's 58 task problem

CT	Inc. Rate	C.V.	Beam Search Solution				Kottas & Lau Solution			Percent Improvement
			# of Stations	Labor Cost	Inc. Cost	CPU	# of Stations	Labor Cost	Inc. Cost	
500	1,5	0,15	4	2000	5,36	282,54	4	2000	17,005	0,58
		0,25	4	2000	8,47	342,24	4	2000	21,548	0,65
	5	0,15	4	2000	4,28	534,32	4	2000	4,37	0,00
		0,25	4	2000	15,53	475,82	4	2000	17,59	0,10
600	1,5	0,15	3	1800	0,41	208,72	3	1800	7,982	0,42
		0,25	3	1800	3,42	165,44	3	1800	5,965	0,14
	5	0,15	3	1800	0,31	211,19	3	1800	5,642	0,30
		0,25	3	1800	12,15	157,25	3	1800	16,769	0,25
700	1,5	0,15	3	2100	6,24	159,22	3	2100	5,216	-0,05
		0,25	3	2100	5,63	140,66	3	2100	9,979	0,21
	5	0,15	3	2100	3,09	200,37	3	2100	6,59	0,17
		0,25	3	2100	21,4	159,72	3	2100	7,197	-0,67

Table 5.6: Results for Tonge's 70 task problem

CT	Inc. Rate	C.V.	Beam Search Solution				Kottas & Lau Solution			Percent Improvement
			# of Stations	Labor Cost	Inc. Cost	CPU	# of Stations	Labor Cost	Inc. Cost	
800	1,5	0,15	5	4000	6,04	1085,98	5	4000	7,357	0,03
		0,25	5	4000	44,193	1772	6	4800	43,513	19,76
	5	0,15	5	4000	30,83	1268,01	5	4000	91,742	1,51
		0,25	5	4000	174,48	790,49	5	4000	297,41	2,94
1000	1,5	0,15	4	4000	3,69	416,66	4	4000	10,003	0,16
		0,25	4	4000	16,54	979,66	4	4000	28,209	0,29
	5	0,15	4	4000	50,62	412,05	4	4000	95,626	1,11
		0,25	4	4000	100	677,18	4	4000	136,758	0,90
1200	1,5	0,15	4	4800	1,93	387,55	4	4800	1,324	-0,01
		0,25	4	4800	2,06	403,15	4	4800	16,599	0,30
	5	0,15	3	3600	285,45	319,44	4	4800	23,264	24,14
		0,25	4	4800	112,99	484,99	4	4800	131,003	0,37

5.2 Computational Results for ULBP

In this section we present the results of some problems for U-type configuration. These problems are the same problems that solved for straight line balancing problem. As previously mentioned in chapter 2, there is no available method in literature for the cost-based evaluation of U-type assembly systems with stochastic process times. For this reason, we cannot compare our results to those of any other algorithm. Moreover, exact calculation of incompletion probabilities is very difficult, because the incompletion probability tree expands very quickly and two units moving in different directions at a station makes incompletion probability of one unit dependent on the other unit. Therefore, we proposed a heuristic method for estimating expected cost of a U-line design. The details of the proposed method are presented in chapter 3. However, we need to be confident that, the proposed heuristic cost estimation method estimates within a desired accuracy. For this reason several designs are evaluated under the proposed estimation method and the cost estimate for these designs are compared to cost estimates obtained by simulating the line designs.

The investigated line designs are simulated until steady state is reached and 10 replications are taken for each line design. A 95 % confidence interval is constructed for the average cost per unit produced by the line and it is checked whether the cost, estimated by proposed method lies within this confidence interval. In Figures 5.1 to 5.4 and Tables 5.7. through 5.8. we present the problem studied and its supporting data as well as the design configurations.

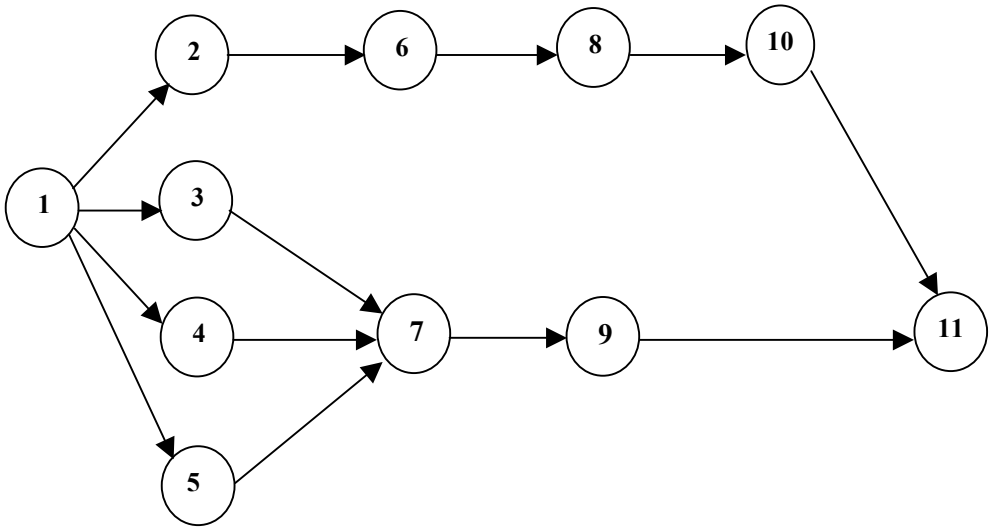


Figure 5.1: Precedence diagram for the test problems.

Table 5.7. Task means: (T_i *for the* i^{th} *task*)

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}
6	2	4	9	2	2	3	6	5	5	3

Table 5.8: Task variances: (σ^2_i *for the* i^{th} *task*)

σ^2_1	σ^2_2	σ^2_3	σ^2_4	σ^2_5	σ^2_6	σ^2_7	σ^2_8	σ^2_9	σ^2_{10}	σ^2_{11}
1.2	0.4	1	5	0.4	0.4	0.6	1.2	1	1	1.8

1 11	2 9	3, 6	4, 5	7 10	8

Figure 5.2: Configuration of 1st design.

1 11, 9	3, 4	2, 5, 6 10	7, 8

Figure 5.3: Configuration of 2nd design

1, 4 11	2, 3 10, 8	5, 6 9, 7

Figure 5.4: Configuration of 3rd design

Simulation mean, standard deviation and confidence interval for all three designs as well as the result of proposed heuristic estimation is presented in Table 5.9. The results indicate that in all of the three designs evaluated, cost estimate by proposed heuristic method lies within the *95%* confidence interval obtained by simulating the lines. This suggests that the proposed heuristic finds very good estimates of the incompletion cost of a U-line. Confidence interval upper and lower limits as well as heuristic solution are plotted in Figure 5.5. for all three designs.

Table 5.9: Comparison of simulation results and proposed heuristic results

	Simulation Mean	Simulation St. Dev.	CI Lower Bound	CI Upper Bound	Heuristic Solution
Design 1	18	12.85	17.2	18.79	18.1
Design 2	7.27	5.26	6.98	7.56	7.3
Design 3	2.45	5	2.27	2.63	2.3

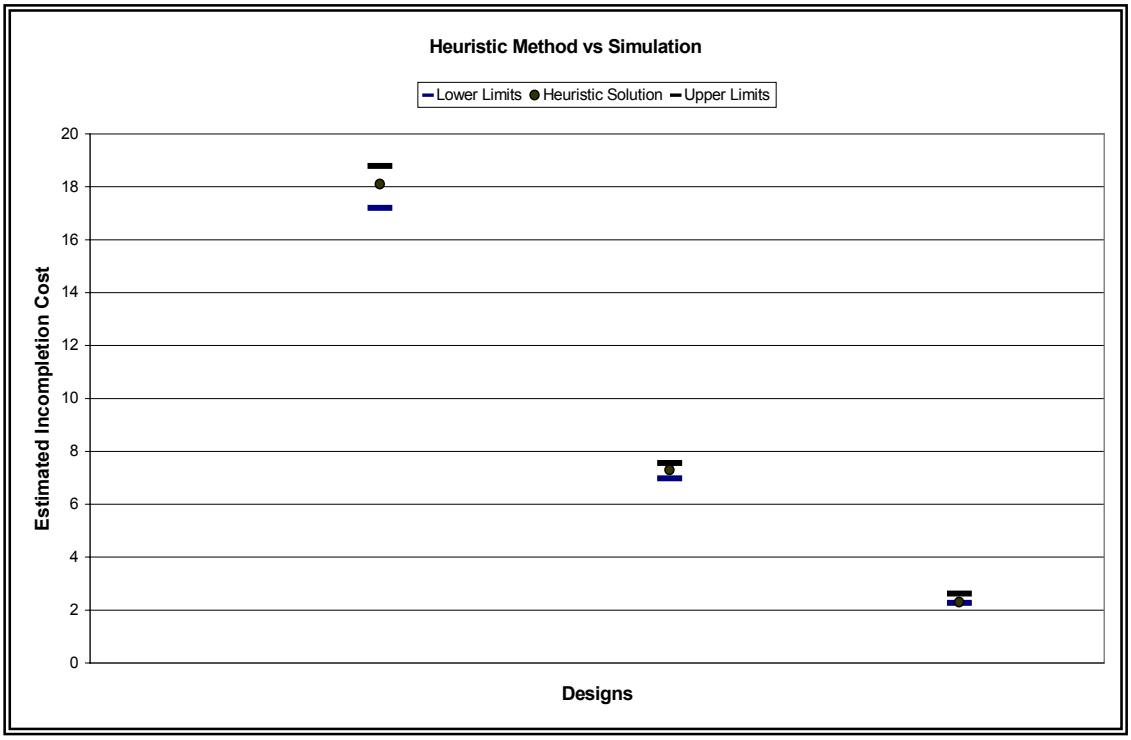


Figure 5.5: Confidence interval and heuristic solution for test problems.

In Tables 5.10. through 5.15. we present the results found by the proposed heuristic. The total cost of line, when U-line configuration is used to solve a problem instance is very close to total cost of line when straight line configuration is used to solve the same instance in most of the problem instances solved. However, U-line configuration can sometimes reduce the total cost of the line design by reducing the number of stations. For example in Table 5.11. on lines 1 and 2, U-line configuration results in 6 stations whereas for the same instances straight line configuration results in 7 stations as indicated by Table 5.2. This situation was expected since as Miltenburg and Wijngaard (1994) state, the U-type configuration increases the solution space of the feasible designs. This is because at any iteration the number of available tasks for assignment in U-type configuration is never less than number of available tasks for assignment in straight line configuration. Thus, U-type configuration presents more design opportunities than straight line configuration.

In some instances U-type configurations can result having greater number of stations, but this is for reducing the incompletion cost component of the total cost. For these instances the total cost terms are very close in both U-type and straight line configurations.

When generated U-line designs are investigated it is seen that, a good U-line design is the one that avoids incompletion on the units moving in forward direction. This is because the contribution of a unit moving in forward direction to incompletion cost has higher magnitude than that of a unit moving in backward direction. The contribution to total cost is higher when the forward moving unit is incomplete in earlier stations and backward moving unit is incomplete in later stations. Thus two design consideration rules can be identified for U-type configuration. These are:

1. When a station has tasks allocated on both sides (forward and backward), the task assignments on forward side must not make the incompletion probability of any forwardly assigned task exceed some threshold value (say τ_f). This threshold value is an increasing function of the number of station under

consideration. In a likewise manner, task assignments on backward side must not make the incompletion probability of any backwardly assigned task exceed a threshold value (say τ_b with $\tau_f \leq \tau_b$). This threshold value is a decreasing function of the number of station under consideration.

2. When task assignments occupy only one side of the line, the task assignments if on forward side, must not make the incompletion probability of any forwardly assigned task exceed some threshold value (say τ_f). Similarly task assignments on backward side must not make the incompletion probability of any backwardly assigned task exceed a threshold value (say τ_b)

In conclusion it appears that a desirable pattern for stochastic U-line balancing solution is one where the workload on forward direction tends to increase as we move in forward direction and workload on backward direction tends to increase as we move in backward direction. Unfortunately, τ_f and τ_b are two hypothetical functions and a definition of what is a proper range for these functions is vague and very difficult. Thus, one must try several values on some test problems to get a rough idea about how their values should be.

We furthermore compare the results of the line costs found by the proposed heuristics in Table 5.16. The results indicate that U-Line configuration generally finds lower cost designs for the studied problems. However, the reader must be reminded that the expected cost of straight lines are found by an exact method whereas the cost of U-Lines are found by a heuristic method.

Table 5.10: Results for Jackson's 11 task problem

CT	Inc. Rate	C.V.	Beam Search Solution			
			# of Stations	Labor Cost	Inc. Cost	CPU
10	1.5	0.15	5	50	12.4368	0.83
		0.25	6	60	6.588	1.43
	5	0.15	6	60	5.333	1.04
		0.25	7	70	10.082	1.37
15	1.5	0.15	4	60	1.0417	0.5
		0.25	4	60	2.505	0.6
	5	0.15	4	60	0.3596	0.55
		0.25	5	75	0.3811	0.66
20	1.5	0.15	3	60	0.0297	0.39
		0.25	3	60	0.9507	0.33
	5	0.15	3	60	0.099	0.44
		0.25	3	60	2.4936	0.33

Table 5.11: Results for Mitchell's 21 task problem

CT	Inc. Rate	C.V.	Beam Search Solution			
			# of Stations	Labor Cost	Inc. Cost	CPU
20	1.5	0.15	6	120	3.5719	3.13
		0.25	6	120	1.9767	4.17
	5	0.15	7	140	0.8849	3.24
		0.25	7	140	7.4553	4.23
30	1.5	0.15	4	120	1.4495	1.31
		0.25	4	120	5.0194	1.65
	5	0.15	4	120	6.092	1.43
		0.25	5	150	0.7133	2.14
40	1,5	0.15	3	120	0.4089	1.21
		0.25	3	120	2.6319	1.42
	5	0.15	3	120	0.6735	1.49
		0.25	3	120	5.998	1.87

Table 5.12: Results for Sawyer's 30 task problem

CT	Inc. Rate	C.V.	Beam Search Solution			
			# of Stations	Labor Cost	Inc. Cost	CPU
40	1.5	0.15	10	400	6.6705	14.77
		0.25	10	400	4.2613	15.32
	5	0.15	10	400	31.5134	15.33
		0.25	11	440	32.1244	18.4
70	1.5	0.15	5	350	11.6969	4.84
		0.25	5	350	17.3369	6.04
	5	0.15	6	420	1.5376	6.81
		0.25	6	420	6.1645	7.25
100	1.5	0.15	4	400	2.1915	4.12
		0.25	4	400	3.6056	4.55
	5	0.15	4	400	0.189	4.23
		0.25	4	400	3.8451	4.4

Table 5.13: Results for Kilbridge's 45 task problem

CT	Inc. Rate	C.V.	Beam Search Solution			
			# of Stations	Labor Cost	Inc. Cost	CPU
100	1.5	0.15	6	600	18.7853	17.96
		0.25	7	700	8.374	23.34
	5	0.15	7	700	2.6078	19.28
		0.25	7	700	10.0831	24.72
150	1.5	0.15	4	600	5.1685	14.72
		0.25	5	750	4.2915	18.62
	5	0.15	4	600	23.0274	13.51
		0.25	5	750	3.7707	14
200	1.5	0.15	3	600	2.5188	10.71
		0.25	3	600	11.6386	14.78
	5	0.15	3	600	2.4322	10.71
		0.25	4	800	0.7338	12.3

Table 5.14: Results for Warnecke's 58 task problem

CT	Inc. Rate	C.V.	Beam Search Solution			
			# of Stations	Labor Cost	Inc. Cost	CPU
500	1.5	0.15	4	2000	3.5663	19.5
		0.25	4	2000	17.674	21.86
	5	0.15	4	2000	0.9451	15.71
		0.25	4	2000	1.4058	14.83
600	1.5	0.15	3	1800	2.509	16.26
		0.25	3	1800	9.1044	19.99
	5	0.15	3	1800	0.2842	16.48
		0.25	3	1800	2.6897	16.21
700	1.5	0.15	3	2100	0.0023	16.15
		0.25	3	2100	0.6148	17.41
	5	0.15	3	2100	0.011	19.16
		0.25	3	2100	0.555	16.48

Table 5.15: Results for Tonge's 70 task problem

CT	Inc. Rate	C.V.	Beam Search Solution			
			# of Stations	Labor Cost	Inc. Cost	CPU
800	1.5	0.15	5	4000	6.711	51.3
		0.25	5	4000	8.083	58.55
	5	0.15	5	4000	92.09	54.93
		0.25	5	4000	131.1088	66.74
1000	1.5	0.15	4	4000	4.0559	40.54
		0.25	4	4000	3.5821	47.07
	5	0.15	4	4000	96.055	48.44
		0.25	4	4000	87.6336	51.08
1200	1.5	0.15	4	4800	3.8047	44.71
		0.25	4	4800	2.7046	49.54
	5	0.15	3	3600	96.652	43.77
		0.25	3	3600	172.77	52.89

Table 5.16: Cost comparison of different line configurations.

11 Task Problem

CT	Inc. Rate	C.V.	Cost of Line Design	
			Straight	U
10	0.15	0.15	61.96	62.43
		0.25	67.23	66.58
	5	0.15	65.34	65.33
		0.25	88.03	80.08
15	0.15	0.15	59.13	61.04
		0.25	62.75	62.50
	5	0.15	60.93	60.35
		0.25	70.87	75.38
20	0.15	0.15	60.70	60.02
		0.25	61.63	60.95
	5	0.15	60.23	60.09
		0.25	63.42	62.49

21 Task Problem

CT	Inc. Rate	C.V.	Cost of Line Design	
			Straight	U
20	0.15	0.15	140.43	123.57
		0.25	146.88	121.97
	5	0.15	140.65	140.88
		0.25	165.79	147.45
30	0.15	0.15	122.57	121.44
		0.25	125.99	125.01
	5	0.15	150.18	126.09
		0.25	151.52	150.71
40	0.15	0.15	122.24	120.40
		0.25	123.14	122.63
	5	0.15	122.39	120.67
		0.25	160.97	125.99

30 Task Problem

CT	Inc. Rate	C.V.	Cost of Line Design	
			Straight	U
40	0.15	0.15	406.32	406.67
		0.25	467.13	404.26
	5	0.15	441.95	431.51
		0.25	500.53	472.12
70	0.15	0.15	371.06	361.69
		0.25	384.52	367.33
	5	0.15	423.15	421.53
		0.25	431.73	426.16
100	0.15	0.15	404.28	402.19
		0.25	404.84	403.60
	5	0.15	400.40	400.18
		0.25	407.89	403.84

45 Task Problem

CT	Inc. Rate	C.V.	Cost of Line Design	
			Straight	U
100	0.15	0.15	709.69	618.78
		0.25	708.19	708.37
	5	0.15	705.92	702.60
		0.25	715.21	710.08
150	0.15	0.15	608.21	605.16
		0.25	756.27	754.29
	5	0.15	755.00	623.02
		0.25	756.82	753.77
200	0.15	0.15	603.42	602.51
		0.25	618.06	611.63
	5	0.15	609.60	602.43
		0.25	806.70	800.73

58 Task Problem

CT	Inc. Rate	C.V.	Cost of Line Design	
			Straight	U
500	0.15	0.15	2005.36	2003.56
		0.25	2008.47	2017.67
	5	0.15	2004.28	2000.94
		0.25	2015.53	2001.40
600	0.15	0.15	1800.41	1802.50
		0.25	1803.42	1809.10
	5	0.15	1800.31	1800.28
		0.25	1812.15	1802.68
700	0.15	0.15	2106.24	2100.00
		0.25	2105.63	2100.61
	5	0.15	2103.09	2100.01
		0.25	2121.40	2100.55

70 Task Problem

CT	Inc. Rate	C.V.	Cost of Line Design	
			Straight	U
800	0.15	0.15	4006.04	4006.71
		0.25	4044.19	4008.08
	5	0.15	4030.83	4092.09
		0.25	4174.48	4131.10
1000	0.15	0.15	4003.69	4004.05
		0.25	4016.54	4003.58
	5	0.15	4050.62	4096.05
		0.25	4100.00	4087.63
1200	0.15	0.15	4801.93	4803.80
		0.25	4802.06	4802.70
	5	0.15	3885.45	3696.65
		0.25	4912.99	3772.77

5.3 Analysis of Results for ULBP and SLBP

At this point, we need to give some statistical results for the effects of design factors considered. The factors we consider are as follows:

1. Line configuration (straight or U-Line configuration)
2. Task time variability (coefficient of variation)
3. Offline completion rate *r*.

We need to figure out, if one configuration type is better than the other. Thus, we first perform analysis of variance on several sample problems. The experiment to be presented is a single-factor experiment. 11, 21, 30, 45 task problems are solved under both U-Line configuration and straight line configuration. Cycle time, coefficient of variation and offline completion rate levels are given in Tables 5. 17 through 5.20 together with the results . For this case, the experimental variable is the configuration type of design. Each resulting design is simulated in Arena 7.0. We take 5 replications and each replication collects data for 1000 units after the steady state for average cost per unit assembled is reached. Analysis of variance results indicate that with 99% significance, U-Line configuration leads in a lower average cost per unit figure than straight line configuration. This is an important piece of information since, it indicates that by constructing a U-Line rather than straight line average cost per unit assembled can be lowered. When line configurations for larger size problems are studied, it is seen that the main advantage of U-Line configuration stems from its ability to balance the line with fewer number of stations. In 12 of the 72 problem instances, U-Line configuration results in fewer number of stations. However, complete information cannot be inferred from single-factor analysis hence a multi-factor analysis of variance should be performed to obtain useful information on the interaction of several design factors with each other. This will be presented later in this chapter, but before presenting the results we perform another single-factor experiment for analyzing the effects of task time variability on both straight and U-Line designs. In this experiment we again solved Jackson's 11 task problem with cycle time fixed at 10, offline completion rate fixed at 5. We

again make the same number of replications and record the same number of data points. The results are presented in Tables 5.21. and 5.22.

According to analysis of variance results, task time variability indicated by *cv* significantly effects the average cost per unit of the line. This fact indicates that variance reduction in task completion times can save from the cost of assembly line. In 8 of the problem instances solved, setting values of coefficient of variation to its high value rather than low, increased the number of stations by one as compared to *cv* at its low value. This increased the labor cost component of the line. Higher task time variance also increased the incompletion cost component of the total cost because with higher task time variance the incompletion probabilities grew larger. However, as previously mentioned cross effects of this factor with other factors should be analyzed before a statement is made.

The effect of offline completion rate to the average cost per unit assembled is obvious since this term is represented linearly in the objective function. Thus, a stand alone experimentation for this factor is not performed. However, in order to see its effect together with other factors a multi-factor analysis of variance is performed. This experiment is also performed for the Jackson's 11 task problem with cycle time fixed at 10. The number of replications and the data points recorded are same as those of previous experiments. The factors and their levels are presented in Table 5.23.

The results of the three factor analysis of variance in Table 5.24. indicate that all three factors are 99% significant on the average cost of per unit assembled. Lower values of *cv* and *r* result in lower average cost per unit and U-Line configuration has lower cost than straight line configuration. The most effective factor on the cost of line is the task time variability represented by *cv*. Higher variability in task times may require higher number of stations which adds to the labor cost of line. This factor also increases the incompletion cost component of the cost function since higher variance in task completion times result in higher incompletion probabilities. The effect of offline completion rate and line configuration are the second and third most effective factors, respectively. The interaction of these factors also 99% significant. It may be inferred from the

multi-factor analysis that closer a design instance to the origin in Figure 5.6. the lower cost it has

Table 5.17: Analysis of variance for effect of line configuration on line cost

Jackson's 11 task problem, C=10, r=5, cv=0.25					
	Replication Averages (Straight Line)		Replication Averages (U-Line)		
1	86.83		83.28		
2	90.72		79.40		
3	88.23		80.57		
4	89.14		81.41		
5	88.15		79.74		
Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0	$F_{0.99,1,8}$
Configuration	149.49	1	149.49	66.95	11.26
Error	17.87	8	2.23		
Total	167.36	9			

Table 5.18: Analysis of variance for effect of line configuration on line cost

Mithcell's 21 task problem, C=40, r=5, cv=0.15					
	Replication Averages (Straight Line)		Replication Averages (U-Line)		
1	127.99		120.80		
2	129.13		120.57		
3	126.84		120.92		
4	126.02		120.89		
5	125.29		120.71		
Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0	$F_{0.99,1,8}$
Configuration	98.37	1	98.37	82.90	11.26
Error	9.49	8	1.18		
Total	107.86	9			

Table 5.19: Analysis of variance for effect of line configuration on line cost

Sawyer's 30 task problem, C=100, r=5, cv=0.25					
	Replication Averages (Straight Line)		Replication Averages (U-Line)		
1	406.32		403.23		
2	408.61		403.76		
3	406.60		404.28		
4	405.20		404.72		
5	406.90		404.78		
Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0	$F_{0.99,1,8}$
Configuration	16.57	1	16.57	16.98	11.26
Error	7.80	8	0.97		
Total	24.37	9			

Table 5.20: Analysis of variance for effect of line configuration on line cost

Kilbrid's 45 task problem, C=200, r=5, cv=0.15					
	Replication Averages (Straight Line)		Replication Averages (U-Line)		
1	608.06		602.05		
2	610.47		602.77		
3	608.96		601.84		
4	608.02		602.07		
5	608.14		602.22		
Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0	$F_{0.99,1,8}$
Configuration	106.88	1	106.88	175.88	11.26
Error	4.86	8	0.60		
Total		9			

Table 5.21: Analysis of variance for effect of variability on straight line cost

Jackson's 11 task problem, C=10, r=5					
	Replication Averages (Straight Line) CV=0.15		Replication Averages (Straight Line) CV=0.25		
1	65.78		84.98		
2	65.63		83.40		
3	64.89		85.96		
4	65.53		85.53		
5	64.69		82.54		
Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0	$F_{0.99,1,8}$
Effect of *cv*	919.49	1	919.49	780.97	11.26
Error	9.42	8	1.17		
Total	928.91	9			

Table 5.22: Analysis of variance for effect of variability on U-Line cost

Jackson's 11 task problem, C=10, r=5					
	Replication Averages (U-Line) CV=0.15		Replication Averages (U-Line) CV=0.25		
1	65.57		86.08		
2	65.52		83.42		
3	64.87		83.71		
4	65.37		86.14		
5	64.87		83.68		
Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0	$F_{0.99,1,8}$
Effect of *cv*	937.79	1	937.79	926.18	11.26
Error	8.10	8	1.01		
Total	945.89	9			

Table 5.23: Factors and their levels

Factor Name	Coefficient of Variation	Offline Completion Rate	Line Configuration
Level	A	B	C
-	0.15	1.5	U-Line
+	0.25	5	Straight Line

Table 5.24: Analysis of variance of line operating cost for three factors.

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0	$F_{0.99,1,32}$
A	1518.93	1	1518.93	2423.28	7.51
B	1039.68	1	1039.68	1658.70	
C	33.98	1	33.98	54.22	
AB	511.15	1	511.15	815.48	
AC	41.88	1	41.88	66.82	
BC	42.20	1	42.20	67.34	
ABC	31.95	1	31.95	50.97	
Error	20.05	32	0.626		
Total	3239.85	39			

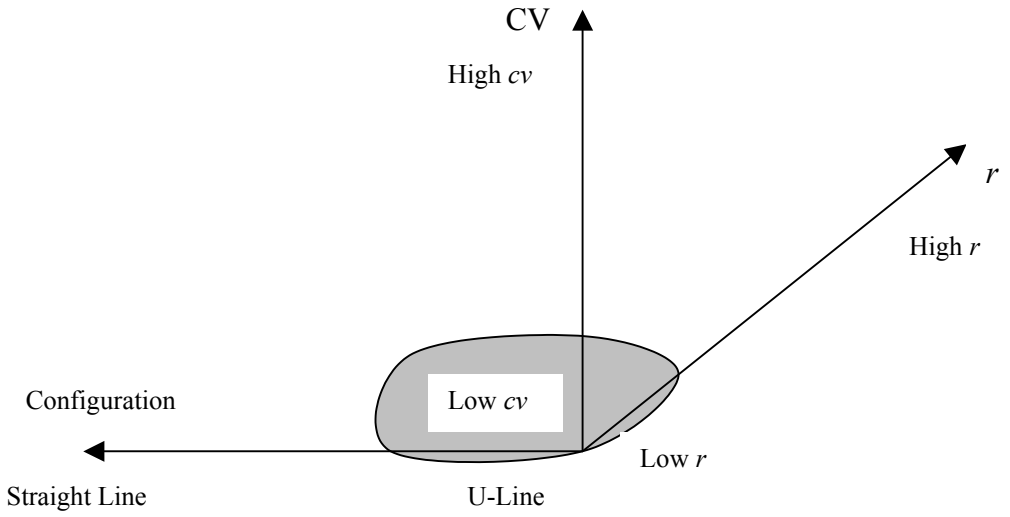


Figure 5.6: The designs closer to the origin have lower line costs.

From a managerial viewpoint, the first thing to consider when balancing an assembly line is the variability in task times. If the variability in task times is high, then whatever the choice of other design factors line operating cost may not be low as desired. The line may produce as many incomplete units as it produces complete units. This is of course undesirable. Thus, variability should be eliminated whenever possible. Use of skilled labor, or training labor for operations highly variable in completion time can be recommended to reduce variability. Furthermore, automation of operations whenever possible can reduce the variability in task completion times. When the variability in task times is at an acceptable level, then by configuring a U-Line design rather than a straight line design we can achieve cost improvements. The results of this study indicate that U-Line configurations can bring reductions in operating cost of line. Unfortunately, in the literature U-Line balancing problem has not been studied in detail up to date. In this thesis we claim and show that U-Lines perform better than straight lines and deserve attention of the people in the manufacturing business.

Chapter 6

Conclusion

In this study, we dealt with the stochastic assembly line balancing problem for both the straight line and U-line configurations. Since the problem is known to be *NP-hard*, there are many heuristic methods developed for the assembly line balancing problem.

This research proposed one such heuristic method for the stochastic assembly line balancing, with the objective of minimizing a line operating cost that consisted of both labor cost and task incompletion cost. We developed a beam search based method for the cost oriented assembly line balancing problem. The idea, key to the core of this research is the concept of marginal desirability. This idea removed the need for rule of thumb methods for closing a station by bringing a rationale behind this decision.

The methodology for evaluating the expected cost of generated designs is taken from Kottas and Lau (1976). This procedure is exact and works by generating and enumerating all possible incompletion tuples.

The solution methodology developed is tested on several problems of varying in size from 11 to 70 tasks. The test problems selected for use in this research are well known and well studied problems. The solution found by the proposed method is compared to that of Kottas and Lau's (1981) algorithm.

For the straight line balancing problem, the results obtained from experimentation on these problems reveals that the proposed heuristic

outperforms its competitor in most of the instances and can improve the solution found by competing algorithm by up to 24 %. The results obtained from this research generally agreed with the following findings of Kottas and Lau (1981) :

1. In order to avoid their incompletion, tasks with high incompletion costs should be performed early in the station.
2. Since there will be a higher probability of incompletion for the tasks assigned to the later part of the cycle time in a station, these tasks should have the lowest possible incompletion costs.
3. Preferably, the last task of a station should have a large mean process time, so that the incompletion probability prior to the last task becomes negligible.
4. Last tasks at a station are responsible for a large percentage of the total expected incompletion cost.
5. The general tendency in assigning tasks to stations is to leave more slack time at the initial stations, since incompletion at these stations generally involve more incompletion cost than that of other stations.

For the U-line balancing problem, analyzing the results led us to the following findings:

1. There is no discernible pattern in leaving idle time at stations. However, there are patterns for workload on forward and backward parts of the station.
2. When a station has task assignments on both forward and backward directions, it is important that a task assigned in forward direction not be incomplete. For this reason, total mean task times of tasks assigned in forward direction must not constitute a high percent of cycle time.
3. The workload on forward direction at a station should be increased as we move from first to last station.
4. The workload on backward direction at a station should be increased as we move from last to first station.

Even though, this thesis concentrated on some design issues, it considered only very known assumptions and settings from literature. However, there are still

several other research issues that deserves attention and needs to be addressed. Below we list the most important of these issues:

1. The method also used the incompletion handling methodology adopted by Kottas and Lau (1973). There are many other policies in the literature and these methodologies worth being studied. Moreover the same incompletion handling policy may not necessarily be used for all the incompletions. For example some serious incompletions may be handled by stopping the entire line while some minor incompletions may be handled after the unit moves down at the end of line. Comparing the effectiveness of such different line operating policies can be a fruitful research area.
2. Workers were assumed to belong to the same skill level and tasks were assumed to be worker independent. Workers may be classified under skilled, semi-skilled and unskilled groups. This way some tasks can only be completed by a worker of a predetermined skill level. Of course, this would make the problem much harder to solve, but also more realistic.
3. Task processing times were assumed to come from normal distribution. Other distributions for task processing times might as well be tried. Thus, the problem can solve instances for which tasks follow a nonsymmetrical distribution. However, such an approach cannot be the extension of this research since normality assumption of task time distributions is crucial to the methods developed here.
4. Maybe the learning effect for the workers may be incorporated to assembly line balancing problem.
5. The mixed model extension of the problem deserves attention for research. A solution methodology developed for the mixed model stochastic assembly line balancing would be more realistic, since in today's production environments rarely a single model product is produced on the assembly lines.
6. In the U-Line balancing problem, workers were assumed to turn between the back and forth of the station in zero time. Travel time for moving within a station may be used in modeling the U-Line balancing problem as Guerrero and Miltenburg (2003) did.

Bibliography

[1] Amen, M., 2000. ' Heuristic Methods for Cost-Oriented Assembly Line Balancing: A survey', *International Journal of Production Economics*, 68 (1), 1-14.

[2] Arcus, A. L., 1966. ' COMSOAL-A Computer Method of Sequencing Operations for Assembly Lines', *International Journal of Operations Research*, 4 (4), 259-277.

[3] Baybars, İ., 1986b. ' A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem', *Management Science*, 32 (8), 909-932.

[4] Bowman, E. H., 1960. 'Assembly Line Balancing by Linear Programming', *Operations Research*, 8, 385-389.

[5] Buxey, G. M., Slack, N. D., and Wild, R., 1973. ' Production Flow Line System Design-A Review' *AIIE Transactions*, 5, 37-48.

[6] Conway, R., Maxwell, W., McClain, J.O., and Thomas, L.J. (1987) The Role of Work-In-Process Inventory in Serial Production Lines. *Operations Research*, 36 (2), 229-241.

[7] Erel, E., Sarin, S.C., 1998. ' A Survey of the Assembly Line Balancing Procedures', *Production Planning and Control*, 9 (5), 414-434.

[8] Erel, E., Sarin, S.C., and Dar-El, E.M., 1999. ' A methodology for solving single-model, stochastic assembly line balancing problem', *Omega-International Journal of Management Science*, 27 (5), 525-535.

[9] Erel, E., Sabuncuoğlu, İ., Aksu, B.A., 2001. ' Balancing of U-Type Assembly Systems Using Simulated Annealing', *International Journal of Production Research*, 39 (13), 3003-3015.

[10] Guerriero, F., and Miltenburg, J. 2003. ' The Stochastic U-line Balancing Problem', *Naval Research Logistics*, 50 (1), 31-57.

[11] 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.

[12] Held, M., and Karp, R. M., 1962. 'A Dynamic Programming Approach to Sequencing Problems', *SIAM*, 10,196-210.

[13] 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.

[14] Hoffmann, T. R., 1992. 'Eureka: A Hybrid System for Assembly Line Balancing', *Management Science*, 38 (1), 39-47.

[15] Ingall, E. J. 1965. 'A Review of Assembly Line Balancing', *Journal of Industrial Engineering*, 16, 244-254.

[16] Jackson, J. R., 1956. 'A Computing Procedure for a Line balancing Problem', *Management Science*, 2 (3), 261-271.

[17] Johnson, R.V., 1973. 'Branch and Bound Algorithm for Assembly Line Balancing and Job-Shop Scheduling', Unpublished Ph.D. Thesis. University of California, Los Angeles.

[18] Johnson, R. V., 1981. 'Assembly Line Balancing Algorithms: Computation Comparisons', *International Journal of Production Research*, 19,277-287.

[19] Johnson, R. V., 1988. 'Optimally Balancing Large Assembly Lines with FABLE', *Management Science*, 34 (2), 240-253.

[20] Kao, E.P.C., 1976.'A Preference Order Dynamic Program for Stochastic Assembly Line Balancing', *Management Science*, 22, 1097-1104.

[21] Kao, E.P.C., and Quayranne, M., 1982. 'On Dynamic Programming Methods for Assembly Line Balancing', *Operations Research*, 30, 375-390.

[22] Kilbridge, M.D., and Wester, L., 1961. 'A Heuristic Method of Assembly Line Balancing', *Journal of Industrial Engineering*, 12 (4), 292-298.

[23] Kilbridge, M.D., and Wester, L., 1962. 'A Review of Analytical Systems of Line Balancing', *Operations Research*, 10 (5), 626-638.

[24] 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.

[25] Kottas, J. F., and Lau, H.S., 1973. 'A Cost Oriented Approach to Stochastic Line Balancing', *AIIE Transactions*, 5,164-171.

[26] Kottas, J. F., and Lau, H.S., 1976. 'A Total Operating Cost Model for Paced Lines with Stochastic Task Times', *AIIE Transactions*, 8, 234-240.

[27] Kottas, J. F., and Lau, H.S., 1981. 'A Stochastic Line Balancing Procedure', *International Journal of Production Research*, 19,177-193.

[28] Magazine, M.J., Doerr, K.H., and Klastorin, T.D., 2000. ' Synchronous Unpaced Flow Lines with Worker Differences and Overtime Cost', *Management Science*, 46 (3), 421-435.

[29] Mansoor, E. M., 1966. 'Improvement on the Gutjahr and Nemhauser algorithm for Line Balancing Problems', *Management Science*, 14, 250-254.

[30] Mastor, A.A., 1970. 'An Experimental Investigation and Comparative Evaluation of Production Line Balancing Techniques', *Management Science*, 16, 728-746.

[31] Matanachai S., Yano, C.A., 2001.' Balancing Mixed Model Assembly Lines to Reduce Work Overload', *IIE Transactions*, 33 (1), 29-42.

[32] Miltenburg, J., 1998. ' Balancing U-Lines in a Multiple U-Line Facility', *European Journal of Operations Research*, 109, 1-23.

[33] Miltenburg, J., 2001-1. ' The Effect of Breakdowns on U-Shaped Production Lines', *International Journal of Production Research*, 38 (2), 353-364.

[34] Miltenburg, J., 2001-2. ' One-piece Flow Manufacturing on U-Shaped Production Lines', *IIE Transactions*, 33, 303-321.

[35] Miltenburg, J., 2001-3. ' U-Shaped Production Lines: A review of Theory and Practice', *International Journal of Production Economics*, 70, 201-214.

[36] Miltenburg, J., and Wijngaard, J., 1994. ' The U-Line Balancing Problem', *Management Science*, 40, 1378-1388.

[37] Monden, Y., 1993. ' Toyota Production System (Norcross, GA: Industrial Engineering and Management Press, Institute of Industrial Engineers).

[38] Moodie, C. L., and Young, H. H., 1965. 'A heuristic Method of Line Balancing for Assumptions of constant of Variable Work Elements', *Journal of Industrial Engineering*, 16, 23-29.

[39] Morton, T. E., and Pentico, D. W., 1993. 'Heuristic Scheduling Systems : with Applications to Production Systems', *Wiley Series in Engineering & Technology Management*.

[40] Nakade, K., and Ohno, K., 1999. ' An Optimal Worker Allocation Problem for a U-Shaped Production Line', *International Journal of Production Economics*, 60-1, 353-358.

[41] Nkasau, M. M., and Leung, K.H., 1995. 'A Stochastic Approach to Stochastic Line Balancing', *International Journal of Production Research*, 33, 975-991.

[42] Nourie, F.J., and Venta, E.R., 1991. 'Finding Optimal Line Balances with OptPack', *Operations Research Letters*, 10, 165-171.

[43] Papadimitriou, C.H., 1982. 'Combinatorial Optimization : Algorithms and Complexity', *Prentice Hall*.

[44] Patterson, J. H., and Albracht, J. J., 1975. 'Assembly-Line Balancing: Zero-One programming with Fibonacci Search', *Operations Research*, 23, 166-174.

[45] Raouf, A., and Tsui, C.L., 1982. 'A New Method for Assembly Line Balancing Having Stochastic Work Elements', *Computers and Industrial Engineering*, 6, 131-148.

[46] Reeve, N.R., and Thomas, W.H., 1973. ' Balancing Stochastic Assembly Lines', *AIIE Transactions*, 5 (3), 223-229.

[47] Sabuncuoğlu, İ., Bayız, M.' Analysis of Reactive Scheduling Problems in a Job Shop Environment', *European Journal of Operational Research*, 126 (3), 567-586.

[48] Sabuncuoğlu, İ., Karabük, S.,1997. ' A Beam Search-Based Algorithm and Evaluation of Scheduling Approaches for Flexible Manufacturing Systems', *IIE Transactions*, 30 (2), 179-191.

[49] Salveson, M.E., 1955. ' The Assembly Line Balancing Problem', *Journal of Industrial Engineering*, (6), 18-25.

[50] Schofield, N. A., 1979. 'Assembly Line Balancing and the Application of Computer Techniques', *Computers and Industrial Engineering*, 3, 53-59.

[51] Scholl, A., and Klein, R.,1999. ' ULINO: Optimally Balancing U-shaped JIT Assembly Lines', *International Journal of Production Research*, (37) 4, 721-736.

[52] Scholl, A., and Klein, R.,1994. 'Combining the Power of FABLE and EUREKA for Assembly Line Balancing - A Bi-directional Branch and Bound Procedure', *Schriften zur Quantitativen Betriebswirtschaftslehre*, 7, TH Darmstadt.

[53] Schrage, L., and Baker, K.R., 1978. ' Dynamic Programming Solution of Sequencing Problems with Precedence Constraints', *Operations Research*, 26 (3), 444-449.

[54] Silverman, F. N., and Carter, J. C., 1986. 'A Cost-Based Methodology for Stochastic Line Balancing with Intermittent Line Stoppages', *Management Science*, 32 (4), 455-463.

[55] Suresh, G., and Sahu, S., 1994. 'Stochastic Assembly Line Balancing Using Simulated Annealing', *International Journal of Production Research*, 32, 1801-1810.

[56] 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.

[57] Wee, T.S., and Magazine, M.J., 1981a. ' An Efficient Branch and Bound Algorithm for Assembly Line Balancing - Part 1: Minimize the Number of Workstations', Working Paper, University of Waterloo, Waterloo, ONT.

[58] Wee, T.S., and Magazine, M.J., 1982. ' Assembly Line Balancing as Generalized Bin Packing', *Operations Research Letters*, 1, 56-58.