FLEXIBLE ASSEMBLY LINE DESIGN PROBLEM WITH FIXED NUMBER OF WORKSTATIONS

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

BY

ŞİRİN BARUTÇUOĞLU

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

JULY 2009

Approval of the thesis:

FLEXIBLE ASSEMBLY LINE DESIGN PROBLEM WITH FIXED NUMBER OF WORKSTATIONS

submitted by **ŞİRİN BARUTÇUOĞLU** in partial fulfillment of requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Nur Evin Özdemirel _____
Head of Department, **Industrial Engineering**

Prof. Dr. Meral Azizoğlu _____
Supervisor, **Industrial Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Ömer Kırca _____
Industrial Engineering Dept, METU

Prof. Dr. Meral Azizoğlu _____
Industrial Engineering Dept, METU

Asst. Prof. Dr. Banu Yüksel Özkaya _____
Industrial Engineering Dept, Hacettepe University

Assoc. Prof. Dr. Canan Sepil _____
Industrial Engineering Dept, METU

Asst. Prof. Dr. İsmail Serdar Bakal _____
Industrial Engineering Dept, METU

Date: _____

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

Name, Last name :

Signature :

ABSTRACT

FLEXIBLE ASSEMBLY LINE DESIGN PROBLEM WITH FIXED NUMBER OF WORKSTATIONS

Barutçuoğlu, Şirin
M.S. Department of Industrial Engineering
Supervisor: Prof. Dr. Meral Azizoğlu

July 2009, 70 pages

In this thesis, we study a Flexible Assembly Line Design problem. We assume the task times and equipment costs are correlated in the sense that for all tasks the cheaper equipment gives no smaller task time. Given the cycle time and number of workstations we aim to find the assignment of tasks and equipments to the workstations that minimizes the total equipment cost. We study a special case of the problem with identical task times. For the general case, we develop a branch and bound algorithm that uses powerful lower bounds and reduction mechanisms. We test the performance of our branch and bound algorithm on randomly generated test problems. The results of our experiments have revealed that we are able to solve large-sized problem instances in reasonable times.

Keywords: Flexible Assembly Lines, Assembly Line Balancing, Branch and Bound Algorithm

ÖZ

SABİT SAYIDA İŞ İSTASYONU İÇEREN ESNEK MONTAJ HATTI TASARIMI PROBLEMİ

Barutçuoğlu, Şirin
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Meral Azizoğlu

Temmuz 2009, 70 sayfa

Bu tezde Esnek Montaj Hattı Tasarımı problemini ele aldık. İş süreleri ve ekipman maliyetlerinin bağlantılı olduğu, işlerin pahalı ekipmanlarla, daha ucuz ekipmanlarla yapıldığından daha uzun sürede yapılamayacağı varsayıldı. Çevrim zamanı ve iş istasyonu sayısı verilmiş iken, işlerin ve ekipmanların iş istasyonlarına toplam ekipman maliyetini en aza indirecek şekilde atanması hedeflendi. Öncelikle problemin özdeş işleri varsayan özel durumu çalışıldı. Genel problem için, güçlü alt limitler ve eleme mekanizmaları kullanan bir dal-sınır algoritması geliştirildi. Dal-sınır algoritmasının performansı rassal olarak yaratılan test problemleri üzerinde değerlendirildi. Deneysel sonuçlar, büyük ölçekli problemlerin önerilen algoritma ile makul sürelerde çözülebildiğini göstermiştir.

Anahtar Kelimeler: Esnek Montaj Hatları, Montaj Hattı Dengeleme, Dal-Sınır Algoritması

To my mom

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my supervisor, Prof. Meral Azizoğlu. She has been more than a supervisor for me. She was always very hearty and friendly as a sister. She did her best to encourage and guide me throughout this study. I would like to underline that she is an excellent person and very important for me. I am very happy that we will be able to work together in the coming years.

I owe thanks to Prof. N. Evin Özdemirel for her sincere support at my hard times. In addition, I would like to thank Assoc. Prof. Canan Sepil for her valuable advices and smiling face. I would also like to thank my examining committee members Prof. Ömer Kırca, Asst. Prof. Banu Yüksel Özkaya, Assoc. Prof. Canan Sepil and Asst. Prof. İsmail Serdar Bakal for their positive attitude and contributions to this study.

I would like to thank my friends Banu Lokman, Büşra Atamer, Melih Çelik and Tülin İnkaya for their warm interest and sympathy. Also, I would also like to thank all my professors for their important contributions. It is a great honor to be a part of METU-IE.

I am grateful to my parents Gülnaz and Şuaip Öztürk for their endless care, support and altruism. I am proud of being their daughter. I would like to expess my love for my brother Göksel Öztürk. I feel very fortunate to have such a sweet brother. I would also like to thank Zeynep and Ömer Barutçuoğlu for their heartiness and goodwill.

My love and my husband, Aras gave a novel touch to my life. I would like to send my deepest thanks to him for his endless love, patience and every thing he added to my life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

An assembly line is a production system, in which different parts are assembled on a product that flows through a sequence of workstations. The workstations are usually connected by a continuous material handling system and a set of assembly tasks is assigned to each workstation. These tasks are indivisible and performed according to some pre-specified restrictions. These restrictions are generally of two types: precedence relations and demand satisfaction. The precedence relations define the technological order such that some tasks can start only after the completion of some other tasks. The demand satisfaction constraint forces the assembly line to deliver a product at the end of each pre-specified period. This period, i.e., the time between two successive product completions, is referred to as cycle time. The cycle time is the reciprocal of the production rate, hence minimizing the cycle time is equivalent to maximizing the production rate.

In Operations Research literature, the decision problem of assigning the assembly tasks to the workstations with respect to some pre-defined objective is called Assembly Line Balancing (ALB) problem. In the literature, basically two types of ALB problems, namely Type 1 and Type 2 ALB problems, are studied. In Type 1 problems, the aim is to minimize the number of workstations given a pre-determined cycle time (hence production rate), whereas in Type 2 problems, the aim is to minimize cycle time (hence maximize production rate) given a fixed number of workstations.

Type 1 problems are usually observed when a new assembly line is to be designed. The purpose is to satisfy the demand with the minimum number of workstations. On the other hand, when the organization wants to produce the

maximum number of products without investing on new machines or expanding the existing ones, Type 2 problems gain significant importance.

In assembly lines, workstations are the places where the resources are assigned and consumed. In traditional assembly lines there is a single resource in each workstation and the resources are identical over all workstations. The single resource is usually represented by a worker together with his/her equipment. On the other hand, flexible assembly lines consider various resources as alternatives for performing each task. The resources may be labor (of different skill) or machinery (of different speed). The resources are usually represented by pieces of equipments where each equipment has a specified cost of assignment and a specified speed to perform each task.

Flexible Assembly Lines are gaining significant value due to their practical importance and theoretical challenge. In practice, to remain competitive in the market, the companies should use Flexible Assembly Lines to achieve high efficiency and respond ever changing customer demands. The automated equipments such as robots or Computer Numerically Controlled machines can offer shorter task times as well as more complex and precise assembly tasks.

Theoretically, the analysis of the Flexible Assembly Lines is challenging due to the complexity brought by equipment alternatives. The alternatives add selection decisions to the task assignment decisions of the traditional lines. The equipment selection decisions have long term impacts as an equipment is usually purchased at high prices. The associated problems are referred to as Flexible Assembly Line Design problems and they usually aim to minimize total equipment cost.

Despite its practical and theoretical importance, the research on Flexible Assembly Line Design problems is quite scarce. The existing literature assumes a limit on the cycle time, but not on the number of workstations. Their

objective is to minimize the total equipment cost, which is equivalent to minimizing the number of workstations when all equipment costs are equal.

In this study, we consider a Flexible Assembly Line Design problem with specified cycle time and fixed number of workstations. That is, we assume that there is a target production rate and the workstations of the line are already located. In such an environment, we aim to minimize the total equipment cost. Moreover, we assume all the task times either decrease or remain same when more advanced, hence expensive, equipment is used. For example, a Computer Numerically Controlled machine is likely to perform the tasks faster than a conventional machine and it is much more expensive.

The rest of thesis is organized as follows: In Chapter 2, we define our problem, introduce the notation and give the mathematical model. The chapter reviews the related literature and introduces a special case with identical task times. Chapter 3 presents our solution approach together with reduction and bounding mechanisms. In Chapter 4, we give the results of our computational experiment. We conclude in Chapter 5 by pointing out main conclusions and suggestions for future research.

CHAPTER 2

THE PROBLEM DEFINITION

In this chapter we first define our problem and give the mathematical formulation of the problem with general task times. We then introduce and give the mathematical model of the problem in which task times are correlated with the equipment costs. Next, we introduce a special case of the problem with identical task times. Finally, we give a brief review of the related literature.

2.1. PROBLEM STATEMENT

We consider a single product Assembly Line Design Problem with equipment decisions, specified minimum production rate and a fixed number of workstations. Our aim is to minimize total equipment cost over all workstations.

We suppose the processing times of the tasks may differ according to their equipments. We assume that each task requires a single equipment and all equipment alternatives are capable of performing all tasks.

2.1.1. THE GENERAL TASK TIMES MODEL

In this section, we consider the Flexible Assembly Line Design problem with general task times. We state our assumptions, give the notation and provide the mixed integer programming model.
Our assumptions are;

- A single product is assembled on the line.
- The tasks are indivisible.

- There is a predetermined upper limit on the number of workstations.
- The cycle time of the line is given.
- All parameters, i.e., task times, equipment costs, precedence structure, cycle time are known with certainty and are not subject to change, i.e., the system is deterministic and static.
- The task times differ with respect to the equipments. We use the terms task times and processing times interchangeably throughout the thesis.
- The task times do not vary according to the workstations and/or the precedence relations.
- The set of equipment types is given and each equipment type has a specific cost. This unit cost includes purchasing and all operational costs. We use the terms equipment types and equipments interchangeably throughout the thesis.
- The equipment costs do not change with respect to tasks.
- There is no set up time between different tasks.
- All tasks can be performed in all workstations and all equipment types can be assigned to all workstations.
- The number of equipments that can be assigned to a workstation is limited.
- The number of tasks that can be assigned to a workstation is not limited.

Sets:

i: the set of tasks to be completed , i=1,2,…,N

l: the set of equipments (or tools) to perform the tasks, l=1,2,…,L

k: the set of workstations that include the equipments , k=1,2,…,K

Parameters:

CT: cycle time, i.e., maximum time allowed in a workstation

K: number of workstations on the line, i.e., maximum number of workstations that can be used

t_{il}: task time of task i when performed with equipment l

EC_l: cost of equipment l

CP_k: equipment capacity of workstation k

$P = \{(a,b) \mid a \text{ immediately precedes } b\}$

Decision Variables:

$$y_{lk} = \begin{cases} 1, \text{if equipment } l \text{ is assigned on workstation } k \\ 0, \text{otherwise} \end{cases}$$

$$x_{ilk} = \begin{cases} 1, \text{if task i is assigned to equipment } l \text{ and workstation } k \\ 0, \text{otherwise} \end{cases}$$

Mathematical Model:

The objective function minimizes the total equipment cost.

$$\text{Min} \sum_{l=1}^{L} \sum_{k=1}^{K} EC_l y_{lk}$$

Constraint set (1) ensures that each task will be assigned to one equipment type and one workstation.

$$\sum_{l=1}^{L} \sum_{k=1}^{K} x_{ilk} = 1 \qquad \forall i \qquad (1)$$

Constraint set (2) makes sure that if a task is assigned to a workstation, its equipment should also be assigned to that workstation.

$$x_{ilk} \le y_{lk} \qquad \forall i,l,k \qquad (2)$$

Constraint set (3) makes sure that the minimum required production rate is satisfied, that is the cycle time limit is not exceeded.

$$\sum_{l=1}^{L}\sum_{i=1}^{N} t_{il}\, x_{ilk} \le CT \qquad \forall k \qquad (3)$$

Constraint set (4), prevents precedence violation. It guarantees that if task *a* immediately precedes task *b* then task *a* cannot be assigned to a later workstation than task *b*.

$$\sum_{l=1}^{L}\sum_{k=1}^{K} k\, x_{alk} \le \sum_{l=1}^{L}\sum_{k=1}^{K} k\, x_{blk} \qquad \forall (a,b) \text{ such that } a \text{ immediately precedes } b \qquad (4)$$

Constraint set (5) limits the number of equipments assigned to a workstation.

$$\sum_{l=1}^{L} y_{lk} \le CP_k \qquad \forall k \qquad (5)$$

Constraint sets (6) – (7) are the binary assignment constraints.

$$x_{ilk} \in \{0,1\} \qquad \forall i,l,k \qquad (6)$$

$$y_{lk} \in \{0,1\} \qquad \forall l,k \qquad (7)$$

2.1.2. THE CORRELATED TASK TIMES MODEL

In this section, we assume the task times and equipment costs are correlated in the sense that for all tasks the cheaper equipment gives no smaller task time. Hence we consider a special case of the model stated in Section 2.1.1. The motivation behind this assumption is the fact that usually more advanced and faster equipments are more expensive. For example, CNC machines are more expensive than the conventional ones and they usually perform the tasks quicker, at least no slower.

According to our assumption, for two equipments *a* and *b*, $EC_a > EC_b$, implies $t_{ia} \leq t_{ib}$ for all tasks *i*. We hereafter assume that the equipments are indexed such that $EC_1 > EC_2 > ..... > EC_L$, i.e., the first equipment is the most expensive, hence the fastest, equipment.

We now state an important theorem for all optimal solutions.

Theorem 1: In all optimal solutions, at most one equipment is assigned to each workstation.

Proof: Assume the condition stated in above property does not hold and there are *R* equipments assigned to a workstation, say workstation *k*. The total cost of the equipments on workstation *k*, $Z_k = \sum_{l \in S_k} EC_l$ where S_k is the set of equipments assigned to workstation *k*. Assume equipment *r* is the most expensive equipment in S_k. As $EC_r > EC_s$ implies $t_{ir} \leq t_{is}$, it is always possible to process all tasks with equipment *r* and freed the other *R*-1 equipments. This leads to an equipment cost $Z_k' = EC_r$. As $Z_k' < Z_k$, the solution that contradicts with our property cannot be optimal.

□

Theorem 1 implies that , constraint set (5) is always satisfied as long as the equipment capacity of the workstations is greater than or equal to 1. Hence, the right hand side of constraint set (5) can be set to 1. This setting reduces the solution space, thus leads to a stronger formulation. Moreover constraint sets (2) and (3) can be replaced by a single constraint set since at most one equipment exists in each workstation. The resulting constraint becomes;

$$\sum_{i=1}^{N} t_{il} x_{ilk} \leq CT \cdot y_{lk} \qquad \forall l,k \qquad (8)$$

With these modifications, below is the statement of the model with correlated task times.

$$\text{Min} \sum_{l=1}^{L}\sum_{k=1}^{K} EC_l\, y_{lk}$$

s.to

$$\sum_{l=1}^{L}\sum_{k=1}^{K} x_{ilk} = 1 \qquad \forall i \qquad (1)$$

$$\sum_{i=1}^{N} t_{il}\, x_{ilk} \le CT \cdot\ y_{lk} \qquad \forall l,k \qquad (8)$$

$$\sum_{l=1}^{L}\sum_{k=1}^{K} k\, x_{alk} \le \sum_{l=1}^{L}\sum_{k=1}^{K} k\, x_{blk} \qquad \forall (a,b) \text{ such that a precedes b} \qquad (4)$$

$$\sum_{l=1}^{L} y_{lk} \le 1 \qquad \forall k \qquad (9)$$

$$x_{ilk} \in \{0,1\} \qquad \forall i,l,k \qquad (6)$$

$$y_{lk} \in \{0,1\} \qquad \forall l,k \qquad (7)$$

We hereafter refer to our Flexible Assembly Line Design problem with correlated times and costs as P.

2.1.3. AN EXAMPLE PROBLEM

In this section we illustrate a feasible and an optimal solution of P via an 11 tasks and 5 equipments example. We assume the line has four workstations and the required cycle time is 30 time units. The precedence relations between tasks are shown in the following figure.

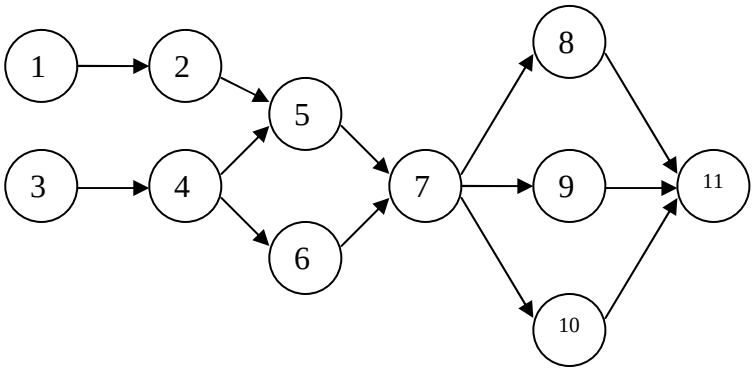

Figure 2.1 Precedence graph of the example problem

The task times depending on the equipment used and equipment costs are given in Table 2.1.

Table 2.1 Task Times and Equipment Costs of the Example Problem

Tasks \ Equipments	1	2	3	4	5
1	7	10	10	13	13
2	6	6	8	12	12
3	6	6	7	8	10
4	8	9	10	11	12
5	6	8	8	9	13
6	6	8	10	12	12
7	6	7	7	7	10
8	8	9	9	10	13
9	8	11	12	12	13
10	8	9	9	11	13
11	6	6	6	9	12
Equipment Costs	400	350	300	250	200

One feasible solution to P is shown in Figure 2.2 .

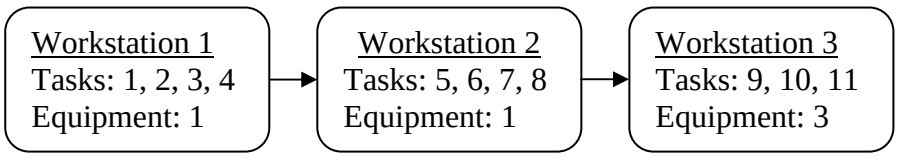

Figure 2.2 A feasible solution of the example problem

Total equipment cost = $EC_1 + EC_1 + EC_3 = 400 + 400 + 300 = 1100$.

The cycle time constraints are satisfied. Specifically,

$t_{11} + t_{21} + t_{31} + t_{41} = 7 + 6 + 6 + 8 = 27 \leq 30$,

$t_{51} + t_{61} + t_{71} + t_{81} = 6 + 6 + 6 + 8 = 26 \leq 30$ and

$t_{93} + t_{10,3} + t_{11,3} = 12 + 9 + 6 = 27 \leq 30$.

The assignments are also precedence feasible.

The following configuration (depicted in Figure 2.3) indicates an optimal solution for P. The solution uses all four workstations.

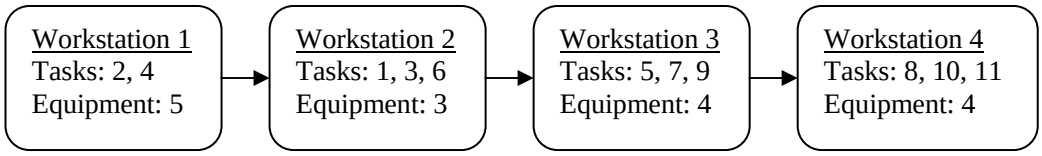

Figure 2.3 An optimal solution of the example problem

The solution is feasible as

$t_{25} + t_{45} = 12 + 12 = 24 \leq 30$,

$t_{13} + t_{33} + t_{63} = 10 + 7 + 10 = 27 \leq 30$,

$t_{54} + t_{74} + t_{94} = 9 + 7 + 12 = 28 \leq 30$,

$t_{84} + t_{10,4} + t_{11,4} = 10 + 11 + 9 = 30 \leq 30$ and the precedence relations hold.

It is optimal with total equipment cost of $EC_5 + EC_3 + EC_4 + EC_4$

$= 200+300+250+250 = 1000.$

2.2. COMPLEXITY OF THE PROBLEM

In this section we show that P is strongly NP-hard through a reduction to the Type 1 ALB problem. Theorem 2 states this result formally.

Theorem 2: P is NP-hard in the strong sense.

Proof: Assume a special case of P, in which $t_{il} = t_i$ for all equipment types l, i.e., the task times are independent of the equipment types. Moreover, assume that $EC_i = EC$, i.e., the equipments are identical. This special case of the problem reduces to the minimization of the number of workstations, i.e., classical Type 1 assembly line balancing problem. If the resulting optimal number of workstations is greater than the available number of workstations, then P is infeasible, otherwise, the limit on the number of workstations is not restrictive. The Type 1 assembly line balancing problem is strongly NP-hard, so is our problem with arbitrary task times and costs.(see Baybars (1986))

□

2.3. A SPECIAL CASE – IDENTICAL TASK TIMES

Suppose $t_{il} = t_l$ for all tasks i and for all equipment types l, i.e., tasks are identical in terms of task times, where t_l is time required for equipment type l to perform any task.

In such a case, n_l, maximum number of tasks an equipment type l can perform in a workstation can be defined as $n_l = \left\lfloor \frac{CT}{t_l} \right\rfloor$.

Using this definition, we formulate a special case of P, which we call Assembly Line Design Problem with Identical Tasks and Correlated Task Times (we refer to this special case as PI) , as follows.

Parameters:

K: maximum number of workstations

EC_l: cost of equipment l

N: number of tasks

n_l: maximum number of tasks an equipment type l can perform in a workstation

Decision Variables:

y_l: number of equipments that will be used from equipment type l

Mathematical Model:

$$\text{Min} \sum_{l=1}^{L} EC_l y_l$$

$$\sum_{l=1}^{L} n_l \cdot y_l \geq N \qquad (10)$$

$$\sum_{l=1}^{L} y_l \leq K \qquad (11)$$

$$y_l \quad \text{integer} \qquad \forall l \qquad (12)$$

Constraint (10) makes sure that all tasks are performed. Constraint (11) ensures there are at most K equipments, since there will be at most one equipment in a workstation (see Theorem 1).

Precedence constraints are not included as we can always define a feasible sequence according to the precedence relations, after finding the optimal solution.

Note that the model has only two constraints in this formulation. So, the optimal LP relaxation of the problem has at most two positive variables. This also means that there can be at most two fractional variables, in the optimal LP relaxed solution.

Below we state some properties of the optimal LP relaxation of PI.

P1) In the optimal solution constraint (10) will be satisfied as strict equality due to the positive objective function coefficients of y_l . Thus, $\sum_{l=1}^{L} n_l \cdot y_l = N$.

Note that, $\frac{EC_l}{n_l}$ gives the cost of performing one task with equipment type *l*. Accordingly, the equipment giving $\text{Min}_l\left\{\frac{EC_l}{n_l}\right\}$ will be favored in the optimal solution. We let *a* be the equipment with $\text{Min}_l\left\{\frac{EC_l}{n_l}\right\}$.

P2) If $\frac{N}{n_a} \le K$ then $y_a = \frac{N}{n_a}$ and $y_i = 0$ for all *i* other than *a*, in the optimal solution.

P3) If $\frac{N}{n_a} > K$, then constraint (11) will be satisfied with strict equality in the optimal solution. Thus, $\sum_{l=1}^{L} y_l = K$.

P4) Let *b* be the equipment type giving $\text{Min}_{l\in B}\left\{\frac{EC_l}{n_l}\right\}$ where $B = \left\{l \mid \frac{N}{n_l} \le K\right\}$, i.e., the set of equipment types that can give a feasible solution alone. $y_b > 0$ in the optimal solution.

Using the results of P2, P3 and P4, we find the optimal relaxed solution with at most *L*-1 trials (number of *b*-*i* pairs). We further reduce the number of trials by P5 and P6.

P5) An equipment type $l \in B$ is eliminated if $\frac{EC_l}{n_l} > \frac{EC_b}{n_b}$. (As b exists in the optimal solution together with an equipment having smaller $\frac{EC_l}{n_l}$ value to reach a smaller objective function value than $EC_b \cdot \frac{N}{n_b}$)

P6) If $n_c = n_d$ and $EC_c \geq EC_d$ for equipment types c and d then y_c cannot take a positive value in the optimal solution.

We now discuss the way that we find the optimal relaxed solution. Using P4 we know, *b*, one of the two equipments that takes positive value in the optimal solution. From P1 and P3, we have the following two equations with two unknowns. P3 implies that it is not feasible to use only the equipment with $\text{Min}_l \left\{ \frac{EC_l}{n_l} \right\}$.

$$n_i \cdot y_i + n_b \cdot y_b = N$$
$$y_i + y_b = K$$

where i is the other equipment that may take positive value. The simultaneous solution of the two equalities yields the following solution values.

$y_b = \frac{N - n_i \cdot K}{n_b - n_i}$ and $y_i = \frac{n_b \cdot K - N}{n_b - n_i}$.

The objective function value with equipments b and i, O_{bi}, is found as,

$$O_{bi} = EC_b \frac{N - n_i \cdot K}{n_b - n_i} + EC_i \frac{n_b \cdot K - N}{n_b - n_i}.$$

Hence the optimal objective function value of the relaxed problem is

$$z = \text{Min}_{i \in \overline{E}} \left\{ \left(EC_b \frac{N - n_i \cdot K}{n_b - n_i} + EC_i \frac{n_b \cdot K - N}{n_b - n_i} \right), EC_b \frac{N}{n_b} \right\}$$

where $\overline{E}$ is the set of the remaining equipments.

If z = $EC_b \frac{N}{n_b}$, i.e., a single equipment is used, then

$y_b^* = \frac{N}{n_b}$, $y_i^* = 0$ for all $i \neq b$.

Otherwise, i.e, if equipments *b* and *i* are used together,

$$y_b^* = \frac{N - n_i \cdot K}{n_b - n_i}, \quad y_i^* = \frac{n_b \cdot K - N}{n_b - n_i}, \quad y_j^* = 0 \text{ for all } j \neq b, i.$$

We now focus on some cases where we find optimal integer solution for the problem. These cases are discussed below.

Case 1. If P6 leaves one equipment, say equipment *b*, then the optimal cost is

$$z = EC_b \left\lceil \frac{N}{n_b} \right\rceil$$

Case 2. If P6 leaves two equipments, say equipment *a* and *b* such that $EC_a < EC_b$. Then $n_a < n_b$.

Note that $\left\lceil \frac{N}{n_a} \right\rceil$ is a valid upper bound on the number of workstations. Hence, we update $K = Min\left\{ K, \left\lceil \frac{N}{n_a} \right\rceil \right\}$.

Let $r \in [1, K]$ be the number of equipments of type *a*. So, the number of equipments of type *b* is $\left\lceil \frac{N - n_a \cdot r}{n_b} \right\rceil$ to perform all *N* tasks on the line. Then $r + \left\lceil \frac{N - n_a \cdot r}{n_b} \right\rceil$ should not exceed the available number of workstations.

Hence, the optimal number of equipments of types *a* and *b* and optimal objective value can be expressed with the following expressions.

$$y_b^* = \left\lceil \frac{N - n_a \cdot y_a^*}{n_b} \right\rceil$$

$$z = \text{Min}_{0 \le r \le K} \left\{ EC_a \cdot r + EC_b \cdot \left\lceil \frac{N - n_a \cdot r}{n_b} \right\rceil \;\middle|\; r + \left\lceil \frac{N - n_a \cdot r}{n_b} \right\rceil \le K \right\}$$

2.4. LITERATURE REVIEW

In this section we give a literature review on Type 2 Assembly Line Balancing (ALB) problems in general and Assembly Line Design problems with equipment decisions in particular.

Although Assembly Line Balancing (ALB) literature is very rich, the research on Flexible Assembly Line Design problem is quite scarce.

Baybars (1986) surveys Type 1 and Type 2 ALB problems. He describes modifications and generalizations of the problems in chronological order. He gives different formulations and proposes exact solution approaches.

Some noteworthy Type 2 ALB problems are due Hackman et al. (1989), Uğurdağ et al. (1997), Rekiek et al. (1999) and Liu et al. (2005). Hackman et al. (1989) propose a heuristic for Type 1 ALB problem. They develop a branch

and bound algorithm that uses the heuristic bounding procedure. They also describe iterative methods to solve Type 2 ALB problem using known upper and lower bounds on the cycle time. To solve Type 2 ALB problems they iteratively solve Type 1 ALB problems. As they mention, the number of iterations can be as large as the difference between upper bound and lower bound on the cycle time.

Uğurdağ et al. (1997) study a bi-criteria Type 2 ALB problem. Their criteria are minimizing the cycle time and balancing the workload. They assume that the processing times on different workstations are equal and the number of workstations is fixed. To solve Type 2 ALB problem they propose a direct approach in place of a sequence of Type 1 ALB problems. They develop a heuristic procedure to find an initial feasible solution and improve the heuristic solution using a simplex-like algorithm.

Rekiek et al. (1999) study a Type 2 ALB problem that balances the workload between the workstations. They assume that processing times on different workstations are equal. In addition to the precedence relations, they include some preference constraints to separate some tasks and group some others. They develop a genetic algorithm based on grouping idea.

Liu et al. (2005) consider a stochastic Type 2 ALB problem with normally distributed and statistically independent task times. They aim to minimize cycle time given a fixed number of workstations and pre-specified cycle time reliability. They propose a heuristic solution procedure.

Rubinovitz and Bukchin (1993), Bukchin and Tzur (2000), Bukchin and Rubinovitz (2002) and Pekin and Azizoğlu (2008) study ALB problems with equipment decisions.

Rubinovitz and Bukchin (1993) study a Type 1 Robotic Assembly Line Balancing Problem (RALB). They assume that each task requires only single

equipment and only one equipment can be assigned to each workstation. They develop a branch and bound algorithm for small sized problem instances and a heuristic method for large sized problem instances.

Bukchin and Tzur (2000) aim to minimize the total equipment cost by considering the pre-specified cycle time, a single equipment requirement for each task and assignment of a single equipment to each workstation. They develop a Branch and Bound algorithm for moderate sized problem instances. In their algorithm, the workstations are opened sequentially, an equipment is assigned once a workstation is opened and then the tasks are selected. For each partial solution a lower bound is computed, that is found by relaxing some of the model constraints. The node with the smallest lower bound is selected for branching. They develop a Branch and Bound based heuristic for large sized problems by modifying their node selection rule.

Bukchin and Rubinovitz (2002) consider Flexible Assembly Line Design problem with station paralleling. They show that adding parallel stations is equivalent to replacing the equipment with a faster one, hence their model is a special case of Bukchin and Tzur (2000)'s model. They adapt the branch and bound algoritm developed by Buckchin and Tzur (2000) for their problem.

Pekin and Azizoğlu (2008) study a bicriteria Flexible Assembly Line Design problem with pre-specified cycle time. Their criteria are the total equipment cost and the number of workstations. They assume multiple equipments can be assigned to each workstation and a single equipment requirement for each task. They develop a branch and bound algorithm to generate all efficient solutions with respect to two criteria.

The most closely related study to ours is Bukchin and Tzur (2000)'s study. Our study differs from their study in the sense that the number of workstations is fixed and task times and equipment costs are correlated.

CHAPTER 3

OUR APPROACH

Recall that our problem P is NP-hard in the strong sense. This justifies the use of an implicit enumeration technique to arrive at an optimal solution. In this study, we propose a branch and bound algorithm to find the optimal assignment of tasks and equipments to the workstations.

The branching schemes designed for classical Assembly Line Balancing (ALB) Problems assign the tasks to the workstations, starting from the first workstation. For the current station, the assignments are considered among the fittable tasks set. A task is called fittable if all its predecessors are assigned either to the current workstation or one of the prior workstations. The current workstation is closed whenever there is no fittable job that can be assigned without exceeding the cycle time.

Our problem differs from the classical branching schemes designed for ALB problems as it includes the equipment decisions.

Through the following theorem we show that the optimal equipment assignment for a given set of assigned tasks is already available.

Theorem 3: Given a set of assigned tasks S_c to a workstation, the optimal equipment is the cheapest equipment, E_c that satisfies $\sum_{i \in S_c} t_{iE_c} \leq CT$.

Proof:

Note from Theorem 1 that there is a single equipment in each workstation. This equipment should be the cheapest one that resides all tasks without exceeding the cycle time en route to minimizing our total equipment cost objective.

□

As the optimal equipment is available for a given set of tasks, we design our branch and bound algorithm based on task assignments but not on equipment assignments. We decide to close a workstation or not to close it even when there are fittable tasks, since the total task time of the tasks assigned to the current workstation changes according to the equipment assigned. When closing the current workstation we determine the optimal equipment considering the set of tasks assigned to this workstation.

In generating the nodes we make use of the following property.

Property 1: If there is a task that fits to the current workstation with the cheapest equipment E_c, then branching to a node that represents closing the current workstation cannot lead to a better solution.

Proof:

Assume a new workstation $k+1$ is opened when there is a task i that fits to the current workstation k with equipment E_c . Assume i is assigned to one of the succeeding workstations $k+a$. ,i can be removed from workstation $k+a$ and placed to workstation k without violating the cycle time constraint (as it fits even with the cheapest equipment) and without increasing the total equipment cost (as the other assignments are kept same). Hence a solution in which i is placed at the current workstation, cannot be worse.

□

We use the result of the above property in designing our branching scheme. We index the tasks based on the precedence structure. In doing so, we give lower numbers to the tasks that appears as predecessors, i.e., if task i is predecessor of task j, then $i<j$. Besides, among the tasks that appear as predecessors the ones with larger number of successors receive higher priority, i.e., we assign lower numbers to these tasks. In order to prevent the duplication of the solutions we always branch to a task with higher index once we are adding to the current workstation. We always add to the current workstation if there is a fittable task

with the cheapest equipment. If there is no fittable task with the cheapest equipment, then we consider the following two branches.

Branch 1. Close the current workstation (Close Branch)

Assign the cheapest possible equipment (see Theorem 2). Let the equipment assigned be *R*.

Branch 2. Not to close the current workstation (Not Close Branch)

Consider the equipments 1,..., $R+m$ where equipment 1 is the most expensive equipment and $R+m$ is the cheapest equipment that has a fittable task. Thus, when a not close branch is considered for the current workstation the optimal equipment should always be more expensive than the optimal equipment of the corresponding close branch.

In our branch and bound algorithm we first evaluate “Close” branch emanating from a node and continue branching. “Not Close” branch is evaluated during backtracking. We give priority to a “Close” branch as it requires cheaper equipment than “Not Close” branch, hence finding a good upper bound earlier is more likely.

The following figure illustrates our branching scheme.

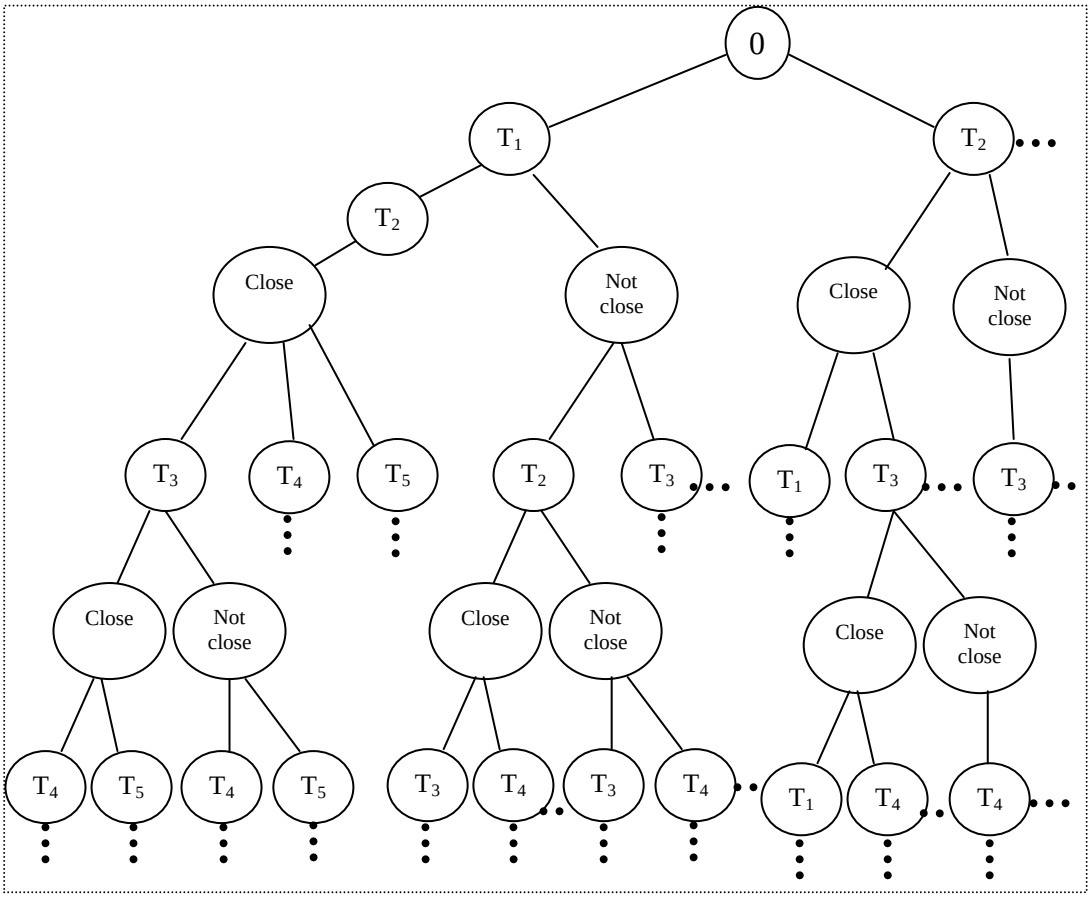

Figure 2.4 The branch and bound tree

For both “Close” and “Not Close” branches, we first check the feasibility of the partial solution in terms of the number of the workstations. In doing so, we use lower bounds on the number of workstations.

- “Close” Branches :

We define a lower bound on the number of the workstations, $K_{min}= \left\lceil \frac{\sum_{i \in U} t_{i1}}{CT} \right\rceil$

where U is the set of unassigned tasks and t_{i1} is the time required to perform task *i* with the most expensive, hence the fastest, equipment.

If $K_{min} + K_1 > K$, then the current partial solution that represents closing workstation K_1 cannot lead to a feasible solution.

If the current solution closes (K-1)st workstation, then the optimal equipment for the next, hence the last, workstation is the largest m that satisfies $\left\lceil \frac{\sum_{i \in U} t_{im}}{CT} \right\rceil = 1$. In such a case, we update the upper bound, i.e., the best known solution if $\sum_{k=1}^{K-1} EC_{E_k} + EC_r < UB$, where E_k is the equipment assigned to workstation k and r is the optimal equipment for the last workstation.

• " Not Close" Branches :

For a "Not Close" branch of workstation K_2 a lower bound on the number of the workstations is,

$$K_{min} = \left\lfloor \frac{\sum_{i \in U} t_{i1}}{CT} \right\rfloor .$$

If $K_{min} + K_2 > K$, then the current partial solution cannot lead to a feasible solution.

We use Property 2 whenever closing a workstation.

Property 2: If task i is assigned to the current workstation, but can be replaced by task j feasibly and $t_{im} \le t_{jm}$ for all m, then the current assignment cannot lead to a unique optimal solution.

Proof:

Assume i is assigned to a later workstation $k+a$. Replacing the workstations of i and j is feasible as j can be assigned to the current workstation when i is removed and i can fit any workstation vacated by j as $t_{im} \le t_{jm}$ for all m. Moreover, such a replacement never increases the number of workstations and

the total equipment cost. This follows that the solution in which j is in the current workstation cannot be worse.

□

While implementing Property 2, we ask: $t_{im} < t_{jm}$ at least for one *m*. This is because if $t_{im} = t_{jm}$ for all *m*, and we fathom a partial solution that represents closing the workstation that includes *i*, then a partial solution that represents closing the workstation that includes j will also be closed.

When closing a workstation, we eliminate some equipment(s) using the results of Properties 3 through 4 stated below.

Property 3: If $\left[\left\lceil \frac{\sum_{i\in U} t_{i1}}{CT} \right\rceil - 1\right] \times EC_{\min} + EC_m \geq UB$, where EC_{min} is the cheapest equipment in the set of remaining equipments, then any further assignment that resides equipment *m* cannot lead to a unique optimal solution.

Proof:

Note that $\left\lceil \frac{\sum_{i\in U} t_{i1}}{CT} \right\rceil$ is a lower bound on the number of workstations. Assume the other workstations are opened with the cheapest equipments and only one workstation is opened with equipment *m*. Such an assignment would have the smallest cost if equipment *m* has to be used at least once. The associated cost is $\left[\left\lceil \frac{\sum_{i\in U} t_{i1}}{CT} \right\rceil - 1\right] \times EC_{\min} + EC_m$. If this cost is no smaller than the smallest known cost, a solution that uses equipment *m* cannot be unique optimal.

□

Property 3 is also used for "Not Close" branches by using $\left\lceil \frac{\sum_{i \in U} t_{i1}}{CT} \right\rceil$ as a lower bound on the number of workstations.

We use the below property in eliminating partial solutions and updating the best known solution, *UB*.

Property 4: If $\left\lceil \frac{\sum_{i \in U} t_{il}}{CT} \right\rceil = 1$ then any further assignment that resides equipments 1,….,*l*-1 cannot be optimal.

Proof:

The minimum cost of completing the partial solution with a single workstation is EC_l. As $EC_l < EC_{l-m}$, a solution that resides equipments 1,….,*l*-1 cannot be optimal.

□

If the condition stated by the above property holds then equipments 1,…,*l*-1 are eliminated. Moreover, *UB* is updated if it is greater than $TC(A) + EC_l$ where $TC(A)$ is the equipment cost for already closed workstations.

Note that if $l = L$, i.e., the cheapest equipment justifies a single workstation, then the node is fathomed. This can be generalized if there is a feasible solution with $\left\lceil \frac{\sum_{i \notin A} t_{i1}}{CT} \right\rceil$ workstations, i.e., minimum number of workstations and cheapest equipment, then it is optimal.

The properties stated above are useful for eliminating relatively expensive equipments. Now we state the properties that enable the elimination of cheaper equipments.

Property 5: If $EC_m > EC_l$ for two equipments *m* and *l*, and $t_{im} = t_{il}$ for all $i \in U$, then any future assignment with equipment *m* cannot be optimal.

Proof:

A solution that resides equipment *m* cannot be optimal as its cost can be reduced by $EC_m - EC_l$ units simply by exchanging equipment *m* with equipment *l*. Note that such an exchange does not affect feasibility as $t_{im} = t_{il}$ for all unassigned tasks.

□

Property 6: Assume K_{left} and N_{left} are number of workstations that are not yet used and number of unassigned tasks, respectively.

If $\left(K_{left} - 1\right) \cdot \left\lfloor \frac{CT}{Min_{i \notin A}\{t_{i1}\}} \right\rfloor + \left\lfloor \frac{CT}{Min_{i \notin A}\{t_{il}\}} \right\rfloor < N_{left}$, then any further assignment that resides equipments *l*,….,*L* cannot be optimal.

Proof:

$\left\lfloor \frac{CT}{Min_i\{t_{i1}\}} \right\rfloor$ is the maximum number of tasks that the most expensive, hence the fastest equipment, can perform in a workstation when the precedence relations are relaxed. Hence, it is an upper bound on the number of tasks that can be performed in a workstation. Similarly, $\left\lfloor \frac{CT}{Min_i\{t_{il}\}} \right\rfloor$ is the maximum number of tasks that equipment *l* can perform in a workstation when the precedence relations are relaxed. Assume the other workstations are opened with the most expensive, hence fastest, equipment and only one workstation is opened with

equipment l. Then, $\left(K_{left}-1\right)\cdot\left\lfloor\frac{CT}{Min_i\{t_{i1}\}}\right\rfloor+\left\lfloor\frac{CT}{Min_i\{t_{il}\}}\right\rfloor$ is an upper bound on the number of tasks that can be performed in K_{left} workstations. If the upper bound on the number of tasks that can be performed is less than the number of unassigned tasks then a solution that resides equipment l cannot be feasible. If equipment l satisfies the condition stated in the property then equipments that are cheaper than l satisfy the condition since $t_{il} \le t_{i(l+m)}$ for all tasks i. This follows that any assignment with equipments $l,\ldots,L$ cannot be optimal.

□

If the condition stated by the above property holds then equipments $l,\ldots,L$ are eliminated. The elimination of cheaper equipments is important since our first and second lower bounds use the cost of the cheapest equipment that is not eliminated. Eliminating cheaper equipments increases the cost of the cheapest remaining equipment, therefore improves our lower bounds' performances.

Whenever closing a workstation, we update K, i.e., the number of workstations to be used. Our initial experiments reveal that an increase in number of workstations significantly increases the solution times of our branch and bound algorithm and all workstations are not necessarily used in the optimal solution. So any reduction in the number of workstations would improve the performance of our branch and bound algorithm.

We now describe the way that we update K.

Note that, $n_{LB}=\left\lfloor\frac{CT}{\mathrm{Max}_l\{\mathrm{Max}_i\{t_{il}\}\}}\right\rfloor$ is a lower bound on the number of tasks that can be performed in a workstation since $\mathrm{Max}_l\{\mathrm{Max}_i\{t_{il}\}\}$ is the maximum of all task times. This follows that $\left\lfloor\frac{N}{n_{LB}}\right\rfloor$ is an upper bound on the number of workstations required for a feasible solution.

Another valid upper bound on the number of workstations is $\left\lfloor \frac{UB}{EC_L} \right\rfloor$. We update K as follows:

$$K = \text{Min}\left\{ K, \left\lfloor \frac{N}{n_{LB}} \right\rfloor, \left\lfloor \frac{UB}{EC_L} \right\rfloor \right\}.$$

We calculate lower bounds for each node that cannot be fathomed by our properties. While computing the lower bounds we use the unassigned tasks, remaining equipments and updated K value.

Lower Bound 1

We consider the special case with identical task times (discussed in Section 2.3) to find a lower bound for our problem P. We let z be an optimal solution to the LP relaxed version of PI. If we set task time t_l on equipment l to $Min_i\{t_{il}\}$, z gives a lower bound to our problem. We solve the LP relaxed version of PI with t_l and let the associated objective function value, z, be LB_1.

We use LB_1 for "Close" branches only. Note that each "Close" branch represents a partial solution with set of assigned tasks A. In this case, the identical tasks problem for tasks $i \notin A$ is solved to find LB_1.

When variability of the task times is low, the optimal solution of P is likely to be close to the optimal solution of PI, hence the associated lower bound becomes closer to the optimal objective function value. We illustrate this by two examples given below. The first one exemplifies a case that is unfavorable for LB_1 whereas the second example illustrates a case where LB_1 works well.

Example 1: Our example problem has 11 tasks and 5 equipment types. The line has 3 workstations and the required cycle time of the line is 30 time units. The task times and equipment costs are given in Table 3.1.

Table 3.1 Task Times and Equipment Costs of the Example Problem 1

Equipments / Tasks	1	2	3	4	5
1	7	10	10	13	13
2	6	6	8	12	12
3	6	6	7	8	10
4	8	9	10	11	12
5	6	8	8	9	13
6	6	8	10	12	12
7	6	7	7	7	10
8	8	9	9	10	13
9	8	11	12	12	13
10	8	9	9	11	13
11	6	6	7	9	12
Equipment Costs	400	350	300	250	200

$$t_l = Min_i\{t_{il}\} = \{6,6,7,7,10\}$$

$$n_l = \left\lfloor \frac{CT}{t_l} \right\rfloor = \left\{ \left\lfloor \frac{30}{6} \right\rfloor, \left\lfloor \frac{30}{6} \right\rfloor, \left\lfloor \frac{30}{7} \right\rfloor, \left\lfloor \frac{30}{7} \right\rfloor, \left\lfloor \frac{30}{10} \right\rfloor \right\} = \{5,5,4,4,3\}$$

Equipments 1 and 3 are eliminated by P6. Equipments 2, 4, 5 are left.

$$B = \left\{ l \mid \frac{N}{n_l} \le K \right\} = \{2,4\}$$

$$\text{Min}_{l \in B} \left\{ \frac{EC_l}{n_l} \right\} = \text{Min} \left\{ \frac{EC_2}{n_2}, \frac{EC_4}{n_4} \right\} = \text{Min} \left\{ \frac{350}{5}, \frac{250}{4} \right\} = 62.5$$

So, b=4. According to P5, there is no need to consider i =2, since $\frac{EC_2}{n_2} > \frac{EC_4}{n4}$.

$$LB_1 = \text{Min}_{i \in \{5\}} \left\{ \left(EC_b \frac{N - n_i \cdot K}{n_b - n_i} + EC_i \frac{n_b \cdot K - N}{n_b - n_i} \right), EC_b \frac{N}{n_b} \right\}$$

$$LB_1 = \text{Min}\left\{\left(EC_4 \frac{N - n_5 \cdot K}{n_4 - n_5} + EC_5 \frac{n_4 \cdot K - N}{n_4 - n_5} \right), EC_4 \frac{N}{n_4} \right\}$$

$$LB_1 = \text{Min}\left\{\left(250 \frac{11 - 3 \cdot 3}{4 - 3} + 200 \frac{4 \cdot 3 - 11}{4 - 3} \right), 250 \frac{11}{4} \right\} = \text{Min}\{700, 687.5\} = 687.5$$

Example 2: Our second example problem also has 11 tasks and 5 equipment types. The line has 2 workstations and the required cycle time is 40 time units. Task times and equipment costs are given in the table below.

Table 3.2 Task Times and Equipment Costs of the Example Problem 2

Equipments / Tasks	1	2	3	4	5
1	7	10	10	13	13
2	6	7	8	12	12
3	6	7	8	11	12
4	8	9	10	11	12
5	6	8	8	11	13
6	6	8	10	12	12
7	6	7	8	11	12
8	8	9	9	11	13
9	8	11	12	12	13
10	8	9	9	11	13
11	6	7	8	11	12
Equipment Costs	400	350	300	250	200

$$t_l = Min_i\{t_{il}\} = \{6, 7, 8, 11, 12\}$$

$$n_l = \left\lfloor \frac{CT}{t_l} \right\rfloor = \left\{ \left\lfloor \frac{40}{6} \right\rfloor, \left\lfloor \frac{40}{7} \right\rfloor, \left\lfloor \frac{40}{8} \right\rfloor, \left\lfloor \frac{40}{11} \right\rfloor, \left\lfloor \frac{40}{12} \right\rfloor \right\} = \{6, 5, 5, 3, 3\}$$

Equipments 2 and 4 are eliminated by P6. Equipments 1, 3, 5 are left.

$B = \left\{ l \mid \frac{N}{n_l} \le K \right\} = \{1\}$. So, b=1.

$$LB_1 = \text{Min}_{i \in \{3,5\}}\left\{\left(EC_b \frac{N - n_i \cdot K}{n_b - n_i} + EC_i \frac{n_b \cdot K - N}{n_b - n_i} \right), EC_b \frac{N}{n_b} \right\}$$

$$LB_1 = \text{Min}\left\{\left(EC_1 \frac{N - n_3 \cdot K}{n_1 - n_3} + EC_1 \frac{n_1 \cdot K - N}{n_1 - n_5} \right), \left(EC_1 \frac{N - n_5 \cdot K}{n_1 - n_5} + EC_1 \frac{n_1 \cdot K - N}{n_1 - n_5} \right), EC_1 \frac{N}{n_1} \right\}$$

$$LB_1 = \text{Min}\left\{\left(400\frac{11-5\cdot 2}{6-5}+300\frac{6\cdot 2-11}{6-5}\right),\left(400\frac{11-3\cdot 2}{6-3}+200\frac{6\cdot 2-11}{6-3}\right),400\frac{11}{6}\right\}$$

$$LB_1 = \text{Min}\{700, 733.33, 733.33\} = 700.$$

Lower Bound 2

Note that $\left\lceil \frac{\sum_i t_{im}}{CT} \right\rceil$ is a valid lower bound on the number of workstations that use equipment m and the cheaper equipments $m+1$, ... , L.

We let $L_m = \left\lceil \frac{\sum_i t_{im}}{CT} \right\rceil$.

This follows $(L_m - 1)\times EC_L + EC_m$ is a valid lower bound on the total cost when equipment m and the cheaper equipments are used.

We let $LB_{TC,m} = (L_m - 1)\times EC_L + EC_m$.

An overall lower bound, LB_2 is available by the following expression.

$LB_2 = \text{Min}_{m \in M'}\{LB_{TC,m}\}$, where $M' = \{m \mid L_m \le K\}$, i.e., alternatives that produce feasible assignments with respect to the number of workstations.

The following example illustrates LB_2 computations.

Example 3:

$CT = 10$ $N = 10$ $K = 4$ $EC_i = 6 - i$ $L = 5$

$\sum_i t_{i1} = 15$, $\sum_i t_{i2} = 20$, $\sum_i t_{i3} = 23$, $\sum_i t_{i4} = 45$, $\sum_i t_{i5} = 52$.

$L_1 = \left\lceil \frac{15}{10} \right\rceil = 2$, $L_1 < 4$ hence there may exist a feasible solution using only equipment 1.

$L_2 = \left\lceil \frac{20}{10} \right\rceil = 2$, $L_2 < 4$ hence there may exist a feasible solution using only equipment 2 and/or equipment 1.

$L_3 = \left\lceil \frac{23}{10} \right\rceil = 3$, $L_3 < 4$ hence there may exist a feasible solution using only equipment 3 and/or equipments 1 and 2.

$L_4 = \left\lceil \frac{45}{10} \right\rceil = 5$, $L_4 > 4$ hence there cannot exist a feasible solution using equipment 4.

$L_5 = \left\lceil \frac{52}{10} \right\rceil = 6$, $L_5 > 4$ hence there cannot exist a feasible solution using equipment 5.

This follows, $M' = \{1, 2, 3\}$.

$LB_{TC,1} = (L_1 - 1) \times EC_5 + EC_1 = 1 + 5 = 6$

$LB_{TC,2} = (L_2 - 1) \times EC_5 + EC_2 = 1 + 4 = 5$

$LB_{TC,3} = (L_3 - 1) \times EC_5 + EC_3 = 2 + 3 = 5$

$LB_2 = \text{Min}_{m \in M'} \{LB_{TC,m}\} = \text{Min}\{6, 5, 5\} = 5$

LB_2 is used as a filtering mechanism as it runs quicker when compared to LB_1 and LB_3. We first calculate LB_2, if it cannot eliminate a partial solution, i.e., $TC_c + LB_2 < UB$, then we calculate LB_1 or LB_3.

As LB_2 is an easy-to-find lower bound, we use it for both “Close” and “Not close” branches.

For “Close” branches, the lower bound is found considering the unassigned tasks and remaining equipments (not eliminated by equipment elimination rules).

Formally,

$L_m = \left\lceil \frac{\sum_{i \notin A} t_{im}}{CT} \right\rceil$ where A is the set of assigned tasks.

$LB_2(A) = \text{Min}_{m \in M''} \left\{ LB_{TC,m} \right\}$ where $M'' = \left\{ m \,|\, L_m(A) \le K \text{ and } k \in \overline{E} \right\}$

and $\overline{E}$ is the set of remaining equipments

For “Not close” branches, the lower bound is again found considering the unassigned tasks and remaining equipments. Lower bound on the number of workstations to be used when equipment *m* and cheaper equipments becomes,

$L_m = \left\lfloor \frac{\sum_{i \notin A} t_{im}}{CT} \right\rfloor$ where A is the set of assigned tasks.

We calculate $LB_2(A)$ as in “Close” branches.

$LB_2(A) = \text{Min}_{m \in M''} \left\{ LB_{TC,m} \right\}$ where $M'' = \left\{ m \,|\, L_m(A) \le K \text{ and } k \in \overline{E} \right\}$

and $\overline{E}$ is the set of remaining equipments.

Lower Bound 3

Recall that Bukchin and Tzur (2000) aim to minimize the total equipment cost by considering the cycle time and precedence relations. There is no restriction on the number of workstations used. Single equipment for each task is assumed and single equipment assignment to each workstation is allowed.

Bukchin and Tzur (2000) propose a lower bound that is obtained by relaxing some of the constraints and surrogating some of them. The relaxed constraints

are the precedence constraints. The surrogate constraint is due to the cycle time. They sum the cycle time constraints over all workstations for all equipment and obtain an aggregate cycle time constraint over all equipments. After the relaxations, the constraint set that ensures at most one equipment for each workstation becomes redundant. Hence all workstations are considered together. The resulting formulation and new decision variables are given below.

$y_l = \sum_{k=1}^{N} y_{lk}$ = total number of type l equipment.

$$x_{il} = \sum_{k=1}^{N} x_{ilk} = \begin{cases} 1 & \text{if task } i \text{ is assigned to equipment } l, \\ 0 & \text{otherwise.} \end{cases}$$

$$\text{Min} \sum_{l=1}^{L} EC_l y_l$$

$$\sum_{l=1}^{L} x_{il} = 1 \qquad \forall i \qquad (13)$$

$$\sum_{i=1}^{N} t_{il} x_{il} \le CT \cdot y_l \qquad \forall l \qquad (14)$$

$$x_{il} \in \{0,1\} \qquad \forall i,l \qquad (15)$$

$$y_l \quad \text{integer} \qquad \forall l \qquad (16)$$

Constraint set (13) ensures that each task is assigned to one equipment. Constraint set (14) is the surrogate cycle time constraint. Constraint sets (15) and (16) are the integrality constraints.

After relaxing constraint set (16), the solution for the resulting problem becomes available by the following expression.

$$x_{il} = \begin{cases} 1 & \text{if } EC_l \cdot t_{il} = \min_j \{EC_j \cdot t_{ij}\}, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i$$

$$y_l = \sum_{i=1}^{N} \frac{t_{il} \cdot x_{il}}{CT} \qquad \forall j$$

Then a lower bound on the original problem is available through the following expression,

$$LB_3 = \sum_{l=1}^{L} EC_l y_l$$

A lower bound on any relaxation of a minimization problem is a valid lower bound on the original problem. Bukchin and Tzur (2000)'s problem assumes no limit on the number of workstations; hence it is a relaxation to our problem. This follows; LB_3 is a valid lower bound to our problem.

We use LB_3 for "Close" branches only. Note that each "Close" branch represents a partial solution with set of assigned tasks *A*. In this case, the relaxed problem for tasks $i \notin A$ and remaining equipments is solved in order to find LB_3.

We illustrate the calculation of LB_1 and LB_3 on two simple examples. In the first one LB_3 works better than LB_1 whereas in the second one LB_1 performs better.

Example 4: Suppose there are 11 tasks and 5 equipments. The line has 3 workstations and the required cycle time is 30 time units. Task times and equipment costs are given in table below.

Table 3.3 Task Times and Equipment Costs of the Example Problem 4

Tasks \ Equipments	1	2	3	4	5
1	7	10	10	13	13
2	6	6	8	12	12
3	6	6	7	8	10
4	8	9	10	11	12
5	6	8	8	9	13
6	6	8	10	12	12
7	6	7	7	7	10
8	8	9	9	10	13
9	8	11	12	12	13
10	8	9	9	11	13
11	6	6	7	9	12
Equipment Costs	400	350	300	250	200

$EC_j \cdot t_{ij}$ values are shown in the table below.

Table 3.4 Equipment Costs x Task Times for the Example Problem 4

Tasks \ Equipments	1	2	3	4	5
1	2800	3500	3000	3250	2600
2	2400	2100	2400	3000	2400
3	2400	2100	2100	2000	2000
4	3200	3150	3000	2750	2400
5	2400	2800	2400	2250	2600
6	2400	2800	3000	3000	2400
7	2400	2450	2100	1750	2000
8	3200	3150	2700	2500	2600
9	3200	3850	3600	3000	2600
10	3200	3150	2700	2750	2600
11	2400	2100	2100	2250	2400

$$x_{il} = \begin{cases} 1 & \text{if } EC_l \cdot t_{il} = \min_j \{EC_j \cdot t_{ij}\}, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i$$

$$y_l = \sum_{i=1}^{N} \frac{t_{il} \cdot x_{il}}{CT} \qquad \forall j$$

Using the equations above, we get:

$x_{15} = 1,\ x_{22} = 1,\ x_{35} = 1,\ x_{45} = 1,\ x_{54} = 1,\ x_{61} = 1,\ x_{74} = 1,\ x_{84} = 1,\ x_{95} = 1,$

$x_{10,5} = 1,\ x_{11,3} = 1.$

$$y_1 = \frac{t_{61} \cdot x_{61}}{CT} = \frac{6}{30} = 0.2$$

$$y_2 = \frac{t_{22} \cdot x_{22}}{CT} = \frac{6}{30} = 0.2$$

$$y_3 = \frac{t_{11,3} \cdot x_{11,3}}{CT} = \frac{6}{30} = 0.2$$

$$y_4 = \frac{t_{54} \cdot x_{54}}{CT} + \frac{t_{74} \cdot x_{74}}{CT} + \frac{t_{84} \cdot x_{84}}{CT} = \frac{9+7+10}{30} = 0.86$$

$$y_5 = \frac{t_{15} \cdot x_{15}}{CT} + \frac{t_{35} \cdot x_{35}}{CT} + \frac{t_{45} \cdot x_{45}}{CT} + \frac{t_{95} \cdot x_{95}}{CT} + \frac{t_{10,5} \cdot x_{10,5}}{CT}$$

$$y_5 = \frac{13+10+12+13+13}{30} = 2.033$$

$$LB_3 = \left\lceil \sum_{l=1}^{L} EC_l y_l \right\rceil = 400 \cdot 0.2 + 350 \cdot 0.2 + 300 \cdot 0.2 + 250 \cdot 0.86 + 200 \cdot 2.033$$

$$LB_3 = 832$$

Recall that for the same example problem LB_1 is found to be 688. For this problem LB_3 works better.

Example 5: There are 11 tasks and 5 equipments. The line has 2 workstations and the required cycle time is 40 time units. Task times and equipment costs are given in the table below.

Table 3.5 Task Times and Equipment Costs of the Example Problem 5

Equipments / Tasks	1	2	3	4	5
1	7	10	10	13	13
2	6	7	8	12	12
3	6	7	8	11	12
4	8	9	10	11	12
5	6	8	8	11	13
6	6	8	10	12	12
7	6	7	8	11	12
8	8	9	9	11	13
9	8	11	12	12	13
10	8	9	9	11	13
11	6	7	8	11	12
Equipment Costs	400	350	300	250	200

$EC_j \cdot t_{ij}$ values are shown in the table below.

Table 3.6 Equipment Costs x Task Times for the Example Problem 5

Tasks \ Equipments	1	2	3	4	5
1	2800	3500	3000	3250	2600
2	2400	2450	2400	3000	2400
3	2400	2450	2400	2750	2400
4	3200	3150	3000	2750	2400
5	2400	2800	2400	2750	2600
6	2400	2800	3000	3000	2400
7	2400	2450	2400	2750	2400
8	3200	3150	2700	2750	2600
9	3200	3850	3600	3000	2600
10	3200	3150	2700	2750	2600
11	2400	2450	2400	2750	2400

Now we get:

$x_{15}=1,\ x_{21}=1,\ x_{31}=1,\ x_{45}=1,\ x_{51}=1,\ x_{61}=1,\ x_{75}=1,\ x_{85}=1,\ x_{95}=1,$

$x_{10,5}=1,\ x_{11,3}=1.$

$$y_1=\frac{t_{21}\cdot x_{21}}{CT}+\frac{t_{31}\cdot x_{31}}{CT}+\frac{t_{51}\cdot x_{51}}{CT}+\frac{t_{61}\cdot x_{61}}{CT}=\frac{6+6+6+6}{40}=0.6$$

$$y_2=0$$

$$y_3=\frac{t_{11,3}\cdot x_{11,3}}{CT}=\frac{8}{40}=0.2$$

$$y_4=0$$

$$y_5=\frac{t_{15}\cdot x_{15}}{CT}+\frac{t_{45}\cdot x_{45}}{CT}+\frac{t_{75}\cdot x_{75}}{CT}+\frac{t_{85}\cdot x_{85}}{CT}+\frac{t_{95}\cdot x_{95}}{CT}+\frac{t_{10,5}\cdot x_{10,5}}{CT}$$

$$y_5=\frac{13+12+12+13+13+13}{40}=1.9$$

$$LB_3=\sum_{l=1}^{L}EC_l y_l=400\cdot 0.6+350\cdot 0+300\cdot 0.2+250\cdot 0+200\cdot 1.9$$

$$LB_3=680$$

Recall that for the same example problem LB_1 is found to be 700. For this problem LB_1 works better.

CHAPTER 4

COMPUTATIONAL EXPERIMENTS

In this chapter we discuss the results of our experiment that is designed to test the performance of our branch and bound algorithm together with the reduction and bounding mechanisms. We take the precedence networks from open literature for varying sizes of tasks. We use precedence graphs included in the data sets of Scholl (1993) at the website http://www.assembly-line-balancing.de/

We generate the task times as follows: The shortest task times, t_{i1} , are generated randomly from discrete uniform distribution between 1 and 6. Then the second shortest task times, i.e., the task times of the second expensive equipment, t_{i2}, are generated randomly as $t_{i2} = r_i \times t_{i1}$, where r_i is uniform between 1 and 1.4. Similarly, we generate $t_{i(l+1)}$ as $r_i \times t_{il}$ using t_{il} and r_i values. Note that to find the task times on each equipment, we generate different r_i values.

We set the number of equipments, L, to 5. Our initial experiments showed that the number of equipments does not have a significant effect on the performance of our algorithm. Hence we try a single value of 5 for L. We generate the following two sets of equipment costs.

Set I	EC_1= 400	EC_2=350	EC_3=300	EC_4=250	EC_5=200
Set II	EC_1= 400	EC_2=375	EC_3=350	EC_4=325	EC_5=300

Note that Set I resides low equipment cost and Set II resides high equipment cost problem instances. Sets I and II also correspond to high and low cost variability cases, respectively.

We use three different sets for cycle times: namely *C1*, *C2* and C*3*. *C1* resides the instances with small cycle time, *C2* has cycle times that is 50 percent more than the cycle time of *C1* and *C3* has cycle times that is two times the cycle time of *C1*.

We set the cycle time of *C1* as $\left\lceil \frac{\sum_i \sum_l t_{il}}{L \times K} \right\rceil$, where L is the number of equipments and K is the number of workstations. $\frac{\sum_i \sum_l t_{il}}{L}$ is the expected total processing time and $\left\lceil \frac{\sum_i \sum_l t_{il}}{L \times K} \right\rceil$ gives the expected cycle time when all K workstations are used.

Hence for each value of N and K we use six combinations of cycle times and equipment cost values.

We vary the number of tasks, N, between 20 and 55, in increments of 5, i.e., we use 8 different N values for *C1* and *C2*. For *C3*, We vary N between 20 and 55, in increments of 5 and between 60 and 80 in increments of 10, i.e., we use 11 different N values We vary the number of workstations, K, between 2 and 8, in increments of 2, i.e., we use 4 different K values.

Hence for *C1* and *C2*, we have $2 \times 8 \times 4 = 64$ each and for *C3*, we have $2 \times 11 \times 4 = 88$ combinations. For each combination we generate 10 problem instances. As a total, we solve 2160 problems.

We code our algorithms in C programming language and implement on Intel Core 2 Duo, 2.33 GHz, 980 MB of RAM PC.

For each problem instance, we set a termination limit of 2 hours. We terminate the execution of the branch and bound algorithm if it does not return a solution

in 2 hours. We record the number of nodes searched and best solution reached at termination.

The rest of the chapter is organized as follows: In Section 4.1., we state the statistics we used to evaluate the performances of the lower bounds and branch and bound algorithm. In Section 4.2, we present our preliminary experiment for selecting the bounding mechanisms to be used in the main experiment. We also test the effectiveness of the reduction mechanisms, by comparing two branch and bound algorithms: one with reduction mechanisms and one without reduction mechanisms. The preliminary experiment includes small-sized problem instances and the main experiment includes large-sized problem instances. In the main experiment we use the bounding mechanisms that are returned as most efficient by the preliminary experiment.

4.1. STATISTICS USED

We use the following statistics to evaluate the performance of our branch and bound algorithm.

- Average computation time in Central Processing Unit (CPU) seconds (Average CPU Time)
- Maximum CPU Time
- Average number of nodes searched
- Maximum number of nodes searched
- Average node number till the optimal solution is found (Average optimality node)
- Maximum optimality node

To evaluate the performance of the lower bounds we use average and maximum deviation of the lower bound from the optimal cost as a ratio of the optimal node. Formally,

$$LB_i D = \text{Percent deviation of the lower bound at the root node} = \frac{OPT - LB_i}{OPT} \times 100,$$

where *OPT* is the total equipment cost and LB_i is the total cost found by lower bound *i* at the root node.

4.2. PRELIMINARY RUNS

Our preliminary runs include the following problem combinations:

N = 20, 25 and 30
K = 2, 4, 6 and 8
CT = *C1* and *C2*
EC = Set I and Set II

We have $3 \times 4 \times 2 \times 2 = 48$ combinations. We generate 10 problem instances for each combination. Hence a total of 480 problems are solved.

The aim of these runs is to select the lower bound(s) to be used for larger sized problem instances. We let BAB_i be the branch and bound algorithm that uses only lower bound *i*. BAB_{ij} is the branch and bound algorithm that uses first lower bound *i* and then lower bound *j* if lower bound *i* cannot fathom. We try BAB_1, BAB_2, BAB_3, BAB_{21} and BAB_{23}. We exclude BAB_{13} from even preliminary runs, as our limited runs had revealed that the reductions obtained in node eliminations could not lead to a reduction in CPU times. As LB_2 is an easy-to-compute bound, it is used as a filtering mechanism before LB_1 and LB_3.

We report the average number of nodes and CPU times for equipment cost sets Set I and Set II, in Table 4.1 and Table 4.2, respectively. The associated worst case, i.e., maximum, results are given in Appendix in Tables A.1 and A.2. The tables also include the results for a branch and bound algorithm that uses no lower bounds, namely BAB_0.

Table 4.1 The branch and bound performances with different lower bounds, Set I

CT	N	K	BAB_1		BAB_2		BAB_3		BAB_{21}		BAB_{23}		BAB_0	
			Average # of nodes	Average CPU time*	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time
1	20	2	184	0.002	184	0.000	184	0.000	184	0.000	184	0.000	184	0.000
		4	2,231	0.002	1,846	0.003	2,160	0.002	1,846	0.003	1,833	0.003	2,231	0.003
		6	15,384	0.017	11,514	0.013	11,957	0.014	11,514	0.016	10,096	0.014	15,384	0.014
		8	53,935	0.069	37,716	0.050	26,841	0.036	37,716	0.052	23,537	0.034	53,984	0.053
	25	2	360	0.000	360	0.000	360	0.000	360	0.000	360	0.000	360	0.002
		4	6,711	0.006	5,713	0.005	6,657	0.006	5,713	0.006	5,713	0.006	6,711	0.006
		6	85,982	0.089	61,461	0.067	56,147	0.061	61,440	0.064	46,424	0.053	86,017	0.077
		8	553,353	0.606	305,193	0.364	184,844	0.223	304,036	0.370	146,151	0.186	566,545	0.514
	30	2	7,759	0.009	7,759	0.011	7,759	0.009	7,759	0.011	7,759	0.010	7,759	0.009
		4	1,804,285	1.983	1,460,657	1.673	1,635,432	1.828	1,460,657	1.669	1,356,489	1.567	1,804,285	1.850
		6	77,409,182	101.663	46,191,821	65.189	24,782,193	35.778	46,191,821	65.494	21,203,737	30.664	77,409,182	89.133
		8	1,582,440,566	2144.956	745,667,886	1133.159	106,654,761	179.806	745,667,886	1146.683	99,169,651	165.456	1,583,406,291	1837.875
2	20	2	249	0.000	249	0.000	249	0.000	249	0.000	249	0.000	249	0.000
		4	669	0.000	598	0.002	669	0.002	598	0.000	598	0.002	669	0.000
		6	4,164	0.006	2,517	0.003	3,874	0.005	2,517	0.003	2,504	0.003	4,164	0.005
		8	11,846	0.016	7,455	0.011	9,154	0.013	7,455	0.011	6,829	0.009	11,875	0.011
	25	2	582	0.002	582	0.000	582	0.002	582	0.002	582	0.002	582	0.000
		4	1,302	0.002	1,239	0.002	1,302	0.002	1,239	0.002	1,239	0.003	1,302	0.002
		6	13,619	0.017	8,653	0.011	13,195	0.014	8,653	0.011	8,623	0.011	13,619	0.014
		8	53,301	0.061	28,161	0.034	37,698	0.045	28,161	0.033	25,300	0.033	53,423	0.052
	30	2	7,964	0.009	7,964	0.011	7,964	0.011	7,964	0.011	7,964	0.011	7,964	0.009
		4	56,918	0.058	49,478	0.052	56,904	0.058	49,478	0.052	49,478	0.052	56,918	0.056
		6	2,558,254	3.352	929,092	1.408	1,711,128	2.344	929,092	1.408	863,895	1.309	2,558,254	3.005
		8	34,816,002	47.613	16,899,066	26.411	12,034,730	18.603	16,899,066	26.445	9,495,028	14.967	34,816,002	41.595

* in seconds

Table 4.2 The branch and bound performances with different lower bounds, Set II

CT	N	K	BAB_1		BAB_2		BAB_3		BAB_{21}		BAB_{23}		BAB_0	
			Average # of nodes	Average CPU time	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time
1	20	2	184	0.000	184	0.000	184	0.000	184	0.000	184	0.000	184	0.000
		4	1,448	0.002	1,398	0.000	1,442	0.002	1,398	0.003	1,394	0.002	1,448	0.002
		6	7,450	0.009	6,663	0.008	6,903	0.009	6,588	0.008	6,104	0.008	7,525	0.008
		8	12,245	0.017	11,033	0.016	9,619	0.014	10,458	0.016	8,545	0.011	12,822	0.016
	25	2	360	0.000	360	0.000	360	0.000	360	0.002	360	0.000	360	0.002
		4	5,508	0.006	5,099	0.006	5,508	0.006	5,099	0.006	5,099	0.003	5,508	0.005
		6	40,251	0.045	35,314	0.041	37,466	0.041	34,227	0.038	31,677	0.036	41,430	0.039
		8	120,315	0.145	101,565	0.127	103,972	0.125	96,557	0.120	83,608	0.105	125,773	0.124
	30	2	7,759	0.011	7,759	0.011	7,759	0.011	7,759	0.010	7,759	0.011	7,759	0.011
		4	194,563	0.214	192,461	0.211	191,228	0.211	192,461	0.211	189,127	0.208	194,563	0.210
		6	15,494,403	17.016	12,498,357	14.192	8,382,070	10.092	12,498,165	14.277	6,196,521	8.041	15,494,595	36.750
		8	120,183,700	193.230	91,603,731	150.148	60,553,130	96.895	91,020,559	151.356	44,241,521	73.758	120,787,008	296.797
2	20	2	200	0.000	200	0.000	200	0.000	200	0.000	200	0.000	200	0.000
		4	520	0.002	519	0.002	520	0.002	519	0.002	519	0.000	520	0.000
		6	1,897	0.003	1,818	0.002	1,891	0.003	1,818	0.000	1,814	0.003	1,897	0.002
		8	5,661	0.006	4,962	0.006	5,149	0.005	4,942	0.006	4,497	0.006	5,681	0.006
	25	2	480	0.000	480	0.002	480	0.002	480	0.000	480	0.000	480	0.000
		4	1,176	0.000	1,175	0.002	1,176	0.002	1,175	0.002	1,175	0.002	1,176	0.000
		6	7,290	0.008	6,800	0.008	7,252	0.006	6,800	0.008	6,762	0.006	7,290	0.006
		8	20,533	0.020	18,429	0.014	19,794	0.020	18,349	0.017	17,656	0.019	20,613	0.019
	30	2	6,930	0.008	6,930	0.008	6,930	0.006	6,930	0.008	6,930	0.006	6,930	0.016
		4	6,729	0.008	6,727	0.008	6,729	0.008	6,727	0.009	6,727	0.006	6,729	0.016
		6	228,676	0.252	227,449	0.249	224,784	0.244	227,449	0.245	223,558	0.244	228,676	0.422
		8	2,967,334	4.033	2,858,296	3.911	2,340,918	3.223	2,847,390	3.898	2,226,443	3.117	2,978,240	5.485

As can be observed from the tables, BAB_2 and BAB_3 perform better than BAB_1. The poor performance of BAB_1 can be attributed to the high variability of the task times where the minimum task time may be too far from many task times. In comparing the performances of BAB_0 and BAB_1, we observe that LB_1 cannot lead to a significant reduction in the average number of nodes searched. When LB_2 is used before LB_1 only a slight reduction in the average number of nodes is observed over BAB_2 and the corresponding average CPU times are a slightly higher. Hence, the savings in the number of nodes is outweighed by the increased CPU times.

The effort spent in calculating the lower bounds is significantly justified in BAB_2 and BAB_3. We can see that there is a considerable reduction in the average number of nodes and CPU times over BAB_0. These reductions are more significant when the number of tasks and workstations are larger. For example, for Set I, *C2*, N=30 and K=8, when BAB_2 is used the average number of nodes and CPU time are reduced from 34,816,002 to 16,899,066 and from 41.595 seconds to 26.411 seconds, respectively. Moreover, when BAB_3 is used the average number of nodes is reduced to 12,034,730 and the resulting average CPU time is 18.603 seconds. However this result cannot be generalized and the branch and bound algorithms do not dominate each other. For example, when Set I, *C2*, N=30 and K=6, the average number of nodes is 929,092 for BAB_2 and 1,711,128 for BAB_3. The corresponding average CPU times are 1.408 and 2.344 seconds.

The average number of nodes and CPU times are higher when the equipment costs are more variable and the cycle time is smaller. Note that Set I and *C1* form the hardest combination. Furthermore, as the number of workstations and number of tasks increase the average CPU times tend to increase exponentially.

Moreover, to see whether it is worth to use our elimination rules or not, we perform an experiment using BAB_2 and BAB_{23}. We design two branch and bound algorithms: one using these mechanisms and one not using them. We

report the average performances in Table 4.3. The associated worst case performances are reported in Appendix, Table A.3.

It can be observed from the tables that our reduction mechanisms improve the performance of both BAB_2 and BAB_{23} considerably. As the number of tasks and workstations increase the reductions in the average number of nodes and CPU times are more significant. For example, when Set I, *C2*, N=30 and K=8, the use of reduction mechanisms in BAB_2 and BAB_{23} reduce average CPU times from 108.027 seconds to 26.411 seconds and from 45.702 to 14.967, respectively.

As a result of our preliminary experiments we see that the branch and bound algorithm, that uses our reduction mechanisms and LB_2 before LB_3 as a filtering mechanism, i.e., BAB_{23} , performs superior.

Table 4.3 The effect of the reduction mechanisms

TC	CT	N	K	BAB_2				BAB_{23}			
				With Reductions		Without Reductions		With Reductions		Without Reductions	
				Average # of nodes	Average CPU time	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time	Average # of nodes	Average CPU time
1	1	20	2	184	0.000	314	0.000	184	0.000	314	0.000
			4	1,846	0.003	3,858	0.002	1,833	0.003	3,624	0.002
			6	11,514	0.013	39,417	0.042	10,096	0.014	25,899	0.030
			8	37,716	0.050	167,381	0.183	23,537	0.034	53,786	0.069
		25	2	360	0.000	944	0.002	360	0.000	944	0.000
			4	5,713	0.005	17,012	0.017	5,713	0.006	16,605	0.017
			6	61,461	0.067	307,892	0.306	46,424	0.053	165,841	0.178
			8	305,193	0.364	2,119,970	2.209	146,151	0.186	616,154	0.689
		30	2	7,759	0.011	8,158	0.011	7,759	0.010	8,158	0.011
			4	1,460,657	1.673	3,712,838	4.156	1,356,489	1.567	3,140,969	3.545
			6	46,191,821	65.189	254,738,869	304.061	21,203,737	30.664	69,873,414	89.520
			8	745,667,886	1133.159	1,717,362,925	4994.184	99,169,651	165.456	253,013,359	377.839
	2	20	2	249	0.000	690	0.000	249	0.000	690	0.000
			4	598	0.002	1,486	0.003	598	0.002	1,486	0.002
			6	2,517	0.003	6,734	0.008	2,504	0.003	6,432	0.008
			8	7,455	0.011	27,981	0.034	6,829	0.009	21,700	0.027
		25	2	582	0.000	1,373	0.002	582	0.002	1,373	0.002
			4	1,239	0.002	4,649	0.006	1,239	0.003	4,649	0.006
			6	8,653	0.011	43,963	0.050	8,623	0.011	42,786	0.050
			8	28,161	0.034	215,971	0.252	25,300	0.033	158,159	0.186
		30	2	7,964	0.011	8,114	0.011	7,964	0.011	8,114	0.009
			4	49,478	0.052	106,857	0.141	49,478	0.052	106,857	0.141
			6	929,092	1.408	2,781,706	3.747	863,895	1.309	2,483,071	3.344
			8	16,899,066	26.411	78,820,775	108.027	9,495,028	14.967	33,119,115	45.702
2	1	20	2	184	0.000	314	0.000	184	0.000	314	0.002
			4	1,398	0.000	2,751	0.002	1,394	0.002	2,726	0.003
			6	6,663	0.008	13,898	0.017	6,104	0.008	11,909	0.013
			8	11,033	0.016	32,388	0.038	8,545	0.011	21,243	0.028
		25	2	360	0.000	944	0.002	360	0.000	944	0.000
			4	5,099	0.006	14,705	0.017	5,099	0.003	14,687	0.016
			6	35,314	0.041	133,748	0.142	31,677	0.036	104,903	0.114
			8	101,565	0.127	520,319	0.563	83,608	0.105	329,519	0.370
		30	2	7,759	0.011	8,158	0.008	7,759	0.011	8,158	0.009
			4	192,461	0.211	337,308	0.445	189,127	0.208	330,113	0.435
			6	12,498,357	14.192	38,916,261	53.594	6,196,521	8.041	14,139,610	19.819
			8	91,603,731	150.148	357,477,391	503.363	44,241,521	73.758	128,657,389	186.195
	2	20	2	200	0.000	690	0.000	200	0.000	690	0.002
			4	519	0.002	1,433	0.002	519	0.000	1,433	0.002
			6	1,818	0.002	4,478	0.003	1,814	0.003	4,445	0.006
			8	4,962	0.006	11,656	0.016	4,497	0.006	10,843	0.014
		25	2	480	0.002	1,373	0.000	480	0.000	1,373	0.002
			4	1,175	0.002	4,112	0.005	1,175	0.002	4,112	0.005
			6	6,800	0.008	30,345	0.036	6,762	0.006	30,211	0.034
			8	18,429	0.014	100,238	0.119	17,656	0.019	90,959	0.108
		30	2	6,930	0.008	8,114	0.006	6,930	0.006	8,114	0.011
			4	6,727	0.008	7,763	0.009	6,727	0.006	7,763	0.011
			6	227,449	0.249	399,909	0.549	223,558	0.244	388,972	0.531
			8	2,858,296	3.911	7,184,902	9.895	2,226,443	3.117	5,002,885	6.919

4.3. MAIN EXPERIMENT

In this section we report on the performance of our branch and bound algorithm that is selected for larger sized problem instances in the previous section, i.e., BAB_{23}. We present the results of our main runs in Tables 4.4 through 4.9. Specifically, we report the results of Set I and *C1* in Table 4.4, Set II and *C1* in Table 4.5, Set I and *C2* in Table 4.6, Set II and *C2* in Table 4.7, Set I and *C3* in Table 4.8, Set II and *C3* in Table 4.9.

As can be observed from the tables, as the number of workstations, *K*, increases, both the average and maximum number of nodes and the CPU times increase significantly. The same result is true for the number of nodes till optimality. An increase in the average and maximum CPU times are more significant when $N \geq 30$. For example, for 45 tasks problem, when equipment costs with high variability, i.e., Set I and low cycle time, *C1* are used, we can see from Table 4.4 that for $K= 2$, the maximum CPU time is less than 2 seconds whereas for $K= 4$, 9 out of 10 instances cannot be solved in 2 hours.

When all other parameter combinations are fixed, the equipment cost set with higher variability, i.e., Set I is more difficult than the equipment cost set with lower variability, i.e., Set II. It can be observed from Tables 4.4 and 4.5 that almost all performance measures are better for Set II. For example in Table 4.4 (Set I) for $N=30$, $K= 8$, the average CPU time is about 165 seconds whereas in Table 4.5 (Set II) the average CPU time is about 74 seconds. Some larger sized problems that could not be solved in 2 hours with equipment cost Set I, can be solved with Set II. The similar observations hold for the results in Tables 4.6 and 4.7 (*C1*) as well as Tables 4.8 and 4.9 (*C2*).

Table 4.4 The performance of our branch and bound algorithm, Set I, *C1*

N	K	BAB_{23}						
		Average # of nodes	Maximum # of nodes	Average # of nodes till optimality	Maximum # of nodes till optimality	Average CPU time	Maximum CPU time	# of unsolved instances
20	**2**	184	206	40	109	0.000	0.000	0
	4	1,833	2,816	1,301	2,498	0.003	0.016	0
	6	10,096	17,052	6,698	11,937	0.014	0.031	0
	8	23,537	63,774	19,023	57,719	0.034	0.078	0
25	**2**	360	449	174	416	0.000	0.000	0
	4	5,713	10,615	4,149	8,242	0.006	0.016	0
	6	46,424	99,060	34,615	98,275	0.053	0.110	0
	8	146,151	231,745	119,067	196,787	0.186	0.312	0
30	**2**	7,759	7,829	312	909	0.010	0.016	0
	4	1,356,489	2,717,212	237,528	2,044,275	1.567	2.859	0
	6	21,203,737	51,471,522	2,038,283	9,827,411	30.664	69.719	0
	8	99,169,651	324,677,324	14,483,554	35,088,570	165.456	526.157	0
35	**2**	4,185	4,426	178	981	0.006	0.016	0
	4	481,000	798,988	7,634	28,196	0.647	1.016	0
	6	8,133,208	16,478,554	1,871,569	15,580,849	12.505	22.860	0
	8	36,644,296	97,915,532	12,695,941	90,724,199	63.542	162.016	0
40	**2**	761,008	798,142	3,504	13,593	0.883	0.984	0
	4	2,137,273,813	4,102,512,134	115,304,039	1,052,036,141	2225.503	4308.656	0
	6	979,471,117	2,305,192,522	522,514,603	1,448,706,608	-	-	-
45	**2**	1,220,106	1,248,056	4,308	25,851	1.734	1.922	0
	4	1,276,412,241	1,628,759,874	60,534,933	590,307,811	7122.458	7200.000	9
50	**2**	12,715	12,738	1,311	9,611	0.031	0.032	0
	4	1,059,097	1,767,876	78,865	263,317	2.298	3.719	0
	6	18,961,026	36,123,596	1,301,004	5,338,409	48.708	100.125	0
	8	383,544,388	624,534,035	188,761,463	583,395,763	956.861	1494.516	0
55	**2**	1,720,396	1,721,338	5,281	26,307	3.111	3.234	0
	4	3,828,543,461	3,972,332,623	97,790,182	971,098,204	-	-	-

Table 4.5 The performance of our branch and bound, Set II , C1

N	K	BAB_{23}						
		Average # of nodes	Maximum # of nodes	Average # of nodes till optimality	Maximum # of nodes till optimality	Average CPU time	Maximum CPU time	# of unsolved instances
20	2	**184**	206	40	109	0	0.000	0
	4	1,394	1,918	1,101	1,660	0.002	0.015	0
	6	6,104	10,597	3,855	6,824	0.008	0.016	0
	8	8,545	17,598	7,464	17,033	0.011	0.031	0
25	2	360	449	174	416	0.000	0.000	0
	4	5,099	9,545	4,215	8,242	0.003	0.015	0
	6	31,677	73,703	25,991	73,120	0.036	0.078	0
	8	83,608	158,033	62,722	102,912	0.105	0.203	0
30	2	7,759	7,829	312	909	0.011	0.016	0
	4	189,127	234,400	17,764	88,458	0.208	0.266	0
	6	6,196,521	20,307,181	909,211	2,415,602	8.041	23.218	0
	8	44,241,521	99,385,259	5,058,174	18,674,260	73.758	161.578	0
35	2	4,185	4,426	178	981	0.005	0.016	0
	4	66,312	137,254	8,831	28,196	0.089	0.172	0
	6	1,909,444	7,309,969	95,955	264,380	2.777	12.234	0
	8	14,106,535	24,655,694	1,948,367	7,686,340	24.292	45.078	0
40	2	761,008	798,142	3,504	13,593	0.889	0.985	0
	4	198,493,181	277,048,491	5,357,191	27,562,308	213.286	298.562	0
	6	2,587,633,583	3,587,212,164	226,690,328	903,030,968	4707.408	7200.000	2
	8	1,915,553,738	4,179,656,967	1,430,097,698	3,263,767,420	-	-	-
45	2	1,220,106	1,248,056	4,308	25,851	1.747	1.937	0
	4	586,506,096	700,091,696	42,635,635	338,822,864	769.953	928.984	0
	6	2,739,286,997	4,244,660,542	762,602,196	3,106,564,459	6728.281	7200.000	9
50	2	12,715	12,738	1,311	9,611	0.031	0.032	0
	4	275,773	411,557	86,740	206,384	0.567	0.859	0
	6	3,652,641	10,183,714	673,134	1,415,391	9.511	28.859	0
	8	20,257,157	28,199,471	9,084,204	21,410,298	53.514	86.797	0
55	2	1,720,396	1,721,338	5,281	26,307	3.109	3.234	0
	4	294,225,893	438,528,472	3,150,300	28,540,014	538.163	807.453	0
	6	2,377,720,417	3,680,191,389	230,970,681	1,109,098,371	5690.820	7200.000	5

Another important conclusion from the tables is that the cycle time has significant influence on the difficulty of the problem. As the cycle time increases, the number of nodes searched, CPU times, number of nodes till optimality decrease considerably. This influence is expected since higher cycle times enable optimal solutions with fewer workstations and the reduction rules for the number of workstations become more effective. When the cycle time is higher we have fewer workstations as more tasks can fit to a workstation. The reduction in CPU times becomes more significant when the number of tasks and workstations are higher. Note from Tables 4.4, 4.6 and 4.8 that when N=50, K= 8 and Set I , for *C1*, *C2* and *C3* the average CPU times are about 957, 22 and 4 seconds, respectively. Theoretically, the generated cycle time values may not return a feasible solution, however in our experiments we always had feasible solutions.

Our experiments reveal that the most difficult combination of equipment costs and cycle time is Set I and *C1* (see Table 4.4) whereas the easiest combination is Set II and *C3* (see Table 4.9). In the most difficult combination set, none of the instances with N=40 and K=6 and 8 can be solved in 2 hours. On the other hand, in the easiest combination set, all of the instances are solved with average CPU times less than 4 seconds for K=6 and 218 seconds for K=8. Moreover, for N=45 and K=4, 9 out of 10 instances cannot be solved in 2 hours in the most difficult combination. However, in the easiest combination set, the average CPU time for instances with N=45 and K=8 is less than 779 seconds.

In general, the average and maximum number of nodes till optimality is very low compared to number of nodes searched especially for larger K and N. For example for N=40, K=6, *TC*=Set II and *CT*=C2 (Table 4.7), the average number of nodes till optimality is 4,918,229 although the average number of nodes searched is 196,985,643. Hence, the problems that cannot be solved until our termination limit of 2 hours are likely to be optimal.

Table 4.6 The performance of our branch and bound algorithm, Set I, C2

N	K	BAB_{23}						
		Average # of nodes	Maximum # of nodes	Average # of nodes till optimality	Maximum # of nodes till optimality	Average CPU time	Maximum CPU time	# of unsolved instances
20	**2**	249	265	22	39	0.000	0.000	0
	4	598	1,160	434	640	0.002	0.016	0
	6	2,504	4,414	2,018	2,851	0.003	0.016	0
	8	6,829	15,216	4,710	9,864	0.009	0.016	0
25	**2**	582	595	34	49	0.002	0.016	0
	4	1,239	2,254	936	2,243	0.003	0.016	0
	6	8,623	15,802	6,184	11,427	0.011	0.016	0
	8	25,300	51,733	18,422	37,180	0.033	0.063	0
30	**2**	7,964	7,984	41	59	0.011	0.016	0
	4	49,478	90,950	1,180	7,908	0.052	0.094	0
	6	863,895	1,469,142	162,093	618,985	1.309	2.094	0
	8	9,495,028	27,119,290	1,290,652	3,968,546	14.967	39.984	0
35	**2**	4,562	4,574	54	69	0.008	0.016	0
	4	22,134	31,454	151	662	0.027	0.047	0
	6	458,913	982,150	7,150	25,955	0.733	1.422	0
	8	5,214,712	10,414,948	461,573	2,754,600	8.516	15.610	0
40	**2**	835,410	835,533	61	79	0.928	1.032	0
	4	14,129,245	34,128,526	988,717	5,762,298	12.888	30.672	0
	6	2,019,244,712	3,361,388,127	32,387,561	252,382,236	3252.142	7200.000	1
	8	1,011,080,577	2,992,833,274	578,651,838	3,566,455,191	6866.002	7200.000	9
45	**2**	1,253,130	1,253,144	76	89	1.685	1.797	0
	4	72,044,248	151,505,592	28,029	75,787	81.994	165.234	0
	6	1,289,842,087	4,076,657,166	9,549,611	55,967,939	6826.067	7200.000	9
50	**2**	12,954	12,966	83	99	0.030	0.032	0
	4	45,963	73,758	6,125	17,019	0.097	0.141	0
	6	1,170,743	2,432,750	72,216	313,713	2.822	5.750	0
	8	8,273,826	13,979,789	792,628	4,525,178	21.955	35.687	0
55	**2**	1,722,108	1,722,126	80	109	3.081	3.109	0
	4	177,501,018	462,548,074	2,568	7,729	314.906	809.437	0
	6	3,213,007,172	3,395,145,272	103,172,272	971,289,432	-	-	-

Table 4.7 The performance of our branch and bound algorithm, Set II , C2

N	K	BAB_{23}						
		Average # of nodes	Maximum # of nodes	Average # of nodes till optimality	Maximum # of nodes till optimality	Average CPU time	Maximum CPU time	# of unsolved instances
20	2	200	206	30	39	0.000	0.000	0
	4	519	686	447	627	0.000	0.000	0
	6	1,814	3,012	1,533	2,754	0.003	0.016	0
	8	4,497	10,098	3,842	9,547	0.006	0.016	0
25	2	480	485	40	49	0.000	0.000	0
	4	1,175	2,254	1,092	2,243	0.002	0.015	0
	6	6,762	12,769	5,859	11,559	0.006	0.016	0
	8	17,656	36,450	13,723	32,104	0.019	0.032	0
30	2	6,930	6,932	50	59	0.006	0.016	0
	4	6,727	7,767	679	1,984	0.006	0.016	0
	6	223,558	379,645	50,533	166,118	0.244	0.422	0
	8	2,226,443	4,025,629	664,545	1,937,899	3.117	5.406	0
35	2	3,840	3,840	60	69	0.003	0.016	0
	4	3,007	3,798	400	1,605	0.003	0.016	0
	6	72,974	137,376	8,378	28,318	0.097	0.156	0
	8	602,487	1,239,386	38,406	158,703	0.938	1.938	0
40	2	626,170	626,174	70	79	0.623	0.625	0
	4	437,555	516,820	71,952	319,440	0.467	0.531	0
	6	196,985,643	256,604,300	4,918,229	22,377,380	210.964	267.562	0
	8	2,189,208,116	3,148,503,241	131,737,543	655,282,641	-	-	-
45	2	939,850	939,859	80	89	1.247	1.250	0
	4	896,472	953,604	46,672	75,987	1.134	1.312	0
	6	581,596,476	700,091,696	44,825,425	338,822,864	759.877	925.515	0
	8	2,512,110,231	3,235,124,539	155,674,335	915,894,201	-	-	-
50	2	10,080	10,080	90	99	0.025	0.031	0
	4	16,357	21,974	5,862	16,759	0.038	0.062	0
	6	274,320	464,098	79,746	248,620	0.563	0.953	0
	8	1,571,207	2,720,300	410,240	875,902	3.852	5.718	0
55	2	1,396,840	1,396,846	100	109	2.681	2.688	0
	4	1,712,744	1,730,518	5,311	23,157	2.778	2.813	0
	6	277,655,237	370,626,826	3,049,650	28,540,014	511.747	683.078	0
	8	3,123,214,463	3,824,320,184	85,005,832	757,549,547	-	-	-

Table 4.8 The performance of our branch and bound algorithm, Set I, C3

N	K	BAB_{23}						
		Average # of nodes	Maximum # of nodes	Average # of nodes till optimality	Maximum # of nodes till optimality	Average CPU time	Maximum CPU time	# of unsolved instances
20	2	200	206	30	39	0.000	0.000	0
	4	461	589	327	421	0.000	0.000	0
	6	938	1,870	753	1,716	0.002	0.015	0
	8	2,494	4,406	1,813	2,843	0.003	0.016	0
25	2	480	485	40	49	0.002	0.015	0
	4	1,092	1,749	875	1,736	0.002	0.016	0
	6	1,846	4,112	1,538	4,053	0.003	0.016	0
	8	7,006	15,970	5,414	14,755	0.008	0.016	0
30	2	6,930	6,932	50	59	0.008	0.016	0
	4	7,706	7,838	286	909	0.011	0.016	0
	6	97,338	177,128	1,297	7,304	0.138	0.250	0
	8	864,446	1,598,226	42,242	177,379	1.327	2.704	0
35	2	3,840	3,840	60	69	0.006	0.016	0
	4	4,079	4,426	296	1,025	0.006	0.016	0
	6	37,133	55,836	1,422	10,722	0.058	0.078	0
	8	479,706	855,452	125,419	782,041	0.780	1.468	0
40	2	626,170	626,174	70	79	0.627	0.641	0
	4	760,494	800,884	2,856	13,593	0.889	0.985	0
	6	24,705,053	39,201,548	265,497	2,443,723	30.805	51.219	0
	8	1,828,664,136	3,339,303,773	36,452,624	252,382,236	2432.555	4454.719	0
45	2	939,850	939,859	80	89	1.236	1.250	0
	4	1,219,353	1,248,056	4,135	25,851	1.733	1.921	0
	6	99,648,079	179,414,684	50,380	185,596	145.717	287.344	0
	8	1,549,548,027	4,069,278,376	14,657,720	62,140,847	6558.592	7200.000	9
50	2	10,080	10,080	90	99	0.025	0.031	0
	4	15,485	19,416	3,403	8,031	0.036	0.047	0
	6	94,148	326,854	30,307	140,834	0.217	0.734	0
	8	1,032,151	1,865,188	128,088	313,713	2.516	4.500	0
55	2	1,396,840	1,396,846	100	109	2.673	2.688	0
	4	1,720,165	1,721,082	5,124	24,895	3.114	3.250	0
	6	361,073,469	725,119,670	6,318,679	58,140,304	753.792	1455.781	0
	8	3,255,425,407	3,402,780,884	118,665,237	971,289,432	-	-	-

Table 4.9 The performance of our branch and bound algorithm, Set II, C3

N	K	BAB_{23}						
		Average # of nodes	Maximum # of nodes	Average # of nodes till optimality	Maximum # of nodes till optimality	Average CPU time	Maximum CPU time	# of unsolved instances
20	2	200	206	30	39	0	0,000	0
	4	467	589	327	421	0,000	0,000	0
	6	666	1.022	516	854	0,002	0,016	0
	8	1.904	3.012	1.556	2.754	0,003	0,016	0
25	2	480	485	40	49	0,000	0,000	0
	4	1.092	1.749	875	1.736	0,003	0,016	0
	6	1.515	3.139	1.364	3.080	0,002	0,016	0
	8	6.289	14.440	5.378	13.471	0,006	0,016	0
30	2	6.930	6.932	50	59	0,008	0,016	0
	4	7.760	7.838	286	909	0,011	0,016	0
	6	59.783	135.964	4.678	25.796	0,066	0,156	0
	8	226.586	379.645	33.728	154.017	0,247	0,437	0
35	2	3.840	3.840	60	69	0,005	0,016	0
	4	4.199	4.426	312	1.185	0,006	0,016	0
	6	17.117	49.960	4.364	30.614	0,025	0,078	0
	8	78.608	137.376	13.820	35.784	0,106	0,172	0
40	2	626.170	626.174	70	79	0,631	0,641	0
	4	760.494	800.884	2.856	13.593	0,892	1,000	0
	6	3.470.317	18.323.548	285.911	2.427.775	3,447	18,563	0
	8	203.499.272	270.437.421	5.609.979	27.562.308	217,066	275,672	0
45	2	939.850	939.859	80	89	1,245	1,250	0
	4	1.219.353	1.248.056	4.135	25.851	1,742	1,937	0
	6	30.678.455	142.104.550	5.474.671	45.221.490	35,845	162,141	0
	8	599.496.467	736.277.596	42.074.076	338.822.864	778,811	926,703	0
50	2	10.080	10.080	90	99	0,023	0,032	0
	4	15.612	19.416	3.403	8.031	0,038	0,047	0
	6	70.359	276.594	23.539	144.038	0,145	0,562	0
	8	294.179	464.098	103.131	248.620	0,608	0,953	0
55	2	1.396.840	1.396.846	100	109	2,684	2,688	0
	4	1.720.307	1.721.314	5.224	25.469	3,127	3,266	0
	6	103.358.423	733.836.494	7.968.953	57.641.562	181,973	1282,141	0
	8	298.004.317	438.529.212	5.900.574	28.540.014	550,572	811,609	0

The average performances of the majority of the instances are close to their maximums. This reveals the consistent behavior of our branch and bound algorithm over all instances. However there is an exception in Table 4.9 for N=55 and K=6 where the average CPU time is about 182 seconds whereas the maximum CPU time is about 1282 seconds. We see that 2 of 10 instances have CPU times of 516 and 1282 seconds and the remaining eight instances are solved in about 2.7 seconds.

Generally, as the number of tasks increases, the average and maximum number of nodes searched, CPU times and number of nodes till optimality increase. This is expected since the number of tasks affects the number of branches, hence the depth of the tree. However, we observe some exceptions in our tables between N=30 and N=35, and between N=45 and N=50. This may be due to random effect or the precedence structure. When precedence relations are fewer, more branches become feasible, hence the problem gets harder.

The flexibility of the precedence graphs is measured by flexibility ratio, FR.

$FR = \dfrac{Number\ of\ zeros\ in\ the\ precedence\ network}{\dfrac{N \times (N-1)}{2}}$, where an entry (i, j) of the precedence network is 1 if task i precedes task j and 0 otherwise. Note that $\dfrac{N \times (N-1)}{2}$ gives the total number of entries in the precedence network. As FR increases problem becomes less restricted, hence more difficult to solve.

We calculate FR ratios for our test problems and tabulate the results in Table 4.10.

Table 4.10 The Flexibility Ratios of test problems

N	FR
20	0.2300
25	0.2833
30	0.5517
35	0.4050
40	0.6435
45	0.5545
50	0.1812
55	0.4471
60	0.3379
70	0.4058
80	0.4399

We find that for N=30 and N=35, FR is 0.5517 and 0.4050, respectively. Thus, when N=35 the instances are more flexible which can explain the exception. Moreover, note that for N=45 and N=50, FR values are 0.5545 and 0.1812, respectively. Hence, we can explain the lower CPU times returned by the larger size problems by their lower flexibility ratios.

In practice, the flexibility ratios of the assembly lines are generally lower. Note from Table 4.10 that all the precedence networks we use have relatively high ratios. Bukchin and Tzur (2000) use FR values that are around 0.1 and 0.4. Hence our experiments consider relatively difficult to solve ones.

We next investigate root node lower bound performances. Tables 4.11, 4.12 and 4.13 report the average and maximum deviations of LB_1, LB_2 and LB_3 from the optimal for *C1*, *C2* and *C3*, respectively.

Table 4.11 The lower bound performances, C1

TC	N	K	%DEV LB_1		%DEV LB_2		%DEV LB_3	
			Average	Maximum	Average	Maximum	Average	Maximum
1	20	2	64.24	66.67	10.61	16.67	14.77	22.41
		4	67.32	80.95	23.49	28.57	9.19	13.94
		6	66.65	75.00	32.06	37.50	10.14	12.30
		8	68.83	80.95	32.39	34.88	8.94	12.98
	25	2	64.24	66.67	10.61	16.67	13.86	21.99
		4	64.16	80.95	24.50	27.27	9.91	12.67
		6	64.40	75.00	34.62	37.50	8.54	10.48
		8	67.55	79.49	32.27	38.46	7.81	11.20
	30	2	64.24	66.67	10.61	16.67	13.50	19.19
		4	67.88	81.82	24.50	27.27	8.80	11.67
		6	64.77	74.19	30.00	35.48	7.13	9.26
		8	68.19	80.49	30.32	31.71	5.58	7.70
	35	2	64.24	66.67	10.61	16.67	12.98	19.91
		4	66.75	80.95	25.89	27.27	10.04	12.81
		6	65.15	74.19	31.63	35.48	7.34	9.25
		8	66.79	80.95	31.68	33.33	6.46	8.27
	40	2	63.64	63.64	9.09	9.09	12.21	14.15
		4	67.62	80.95	23.81	23.81	8.57	10.13
		6	65.02	74.19	33.35	35.48	7.26	8.69
	45	2	63.64	63.64	9.09	9.09	12.47	13.30
		4	71.69	81.82	24.50	27.27	9.45	12.64
	50	2	64.24	66.67	10.61	16.67	13.53	18.50
		4	70.65	81.82	27.06	28.57	11.63	13.71
		6	67.09	75.76	32.19	37.50	9.05	10.57
		8	72.96	80.95	32.76	34.15	7.45	9.04
	55	2	63.64	63.64	9.09	9.09	11.65	13.38
		4	75.32	81.82	24.16	27.27	8.20	10.82
2	20	2	55.87	57.14	4.39	7.14	23.27	27.03
		4	54.71	73.33	11.31	13.33	8.19	12.43
		6	53.90	66.67	17.21	26.76	9.68	15.19
		8	57.04	73.33	15.21	17.39	8.56	11.84
	25	2	55.87	57.14	4.39	7.14	23.74	27.43
		4	49.68	73.33	11.69	13.04	9.25	10.92
		6	50.50	66.67	20.62	27.78	9.06	14.14
		8	55.52	72.41	15.99	26.44	9.39	10.99
	30	2	55.87	57.14	4.39	7.14	22.78	24.98
		4	55.13	73.91	12.08	13.04	7.81	9.97
		6	51.09	66.20	13.96	17.46	7.02	11.74
		8	56.79	73.03	14.41	16.48	7.25	10.12
	35	2	55.87	57.14	4.39	7.14	22.68	25.43
		4	52.70	73.33	12.27	13.04	7.98	10.04
		6	51.31	61.90	15.24	18.75	6.31	11.60
		8	54.38	73.33	14.98	16.48	7.34	8.67
	40	2	55.56	55.56	3.70	3.70	23.43	24.89
		4	54.78	73.33	11.30	13.04	8.85	10.83
		6	49.84	61.90	16.21	17.46	4.86	11.92
		8	56.62	72.73	14.02	15.56	8.19	9.67
	45	2	55.56	55.56	3.70	3.70	23.15	24.62
		4	60.23	73.33	11.50	13.04	8.50	9.70
		6	54.40	66.20	16.10	18.75	5.81	13.33
	50	2	55.87	57.14	4.39	7.14	22.87	25.38
		4	58.14	74.47	13.06	14.89	9.02	11.11
		6	53.64	67.12	15.49	18.75	7.69	13.51
		8	62.61	73.63	15.57	16.48	7.38	10.11
	55	2	55.56	55.56	3.70	3.70	22.62	24.27
		4	65.39	73.91	11.30	13.04	7.69	9.13
		6	59.42	66.67	17.12	18.75	4.46	11.63

Table 4.12 The lower bound performances, C2

TC	N	K	%DEV LB_1		%DEV LB_2		%DEV LB_3	
			Average	Maximum	Average	Maximum	Average	Maximum
1	20	2	49.29	50.00	0.00	0.00	19.17	22.74
		4	59.24	73.33	19.00	21.43	13.99	17.04
		6	67.21	81.82	23.95	28.57	9.48	13.94
		8	64.00	72.41	29.10	31.03	9.76	13.79
	25	2	50.00	50.00	0.00	0.00	19.47	21.99
		4	62.10	73.33	18.86	20.00	13.33	17.78
		6	64.94	81.82	26.23	27.27	10.69	12.67
		8	65.29	71.43	27.88	29.63	8.67	10.85
	30	2	50.00	50.00	0.00	0.00	19.32	20.71
		4	59.24	73.33	18.29	20.00	11.71	15.42
		6	67.05	80.95	23.17	25.00	7.78	10.01
		8	65.29	85.19	27.51	28.57	7.00	8.86
	35	2	50.00	50.00	0.00	0.00	18.82	20.87
		4	57.33	73.33	20.00	20.00	13.27	17.00
		6	66.41	80.95	25.19	27.27	10.01	11.82
		8	64.16	86.21	28.52	31.03	8.14	9.69
	40	2	50.00	50.00	0.00	0.00	19.37	21.31
		4	61.52	73.33	17.71	20.00	11.86	15.47
		6	67.60	80.95	23.77	27.27	8.65	10.14
		8	72.22	85.19	26.72	28.57	6.63	8.74
	45	2	50.00	50.00	0.00	0.00	19.67	20.50
		4	62.10	73.33	18.86	20.00	13.19	15.47
		6	71.77	80.95	24.50	27.27	9.33	12.64
	50	2	50.00	50.00	0.00	0.00	19.24	21.53
		4	67.81	73.33	20.14	21.43	13.71	15.77
		6	70.74	81.82	26.93	27.27	11.90	13.71
		8	69.06	72.41	30.16	31.03	9.67	11.88
	55	2	33.33	33.33	0.00	0.00	19.01	20.60
		4	63.64	63.64	9.09	9.09	11.96	13.64
		6	70.00	75.00	22.50	25.00	9.45	11.30
2	20	2	24.50	25.00	0.00	0.00	12.22	16.75
		4	41.42	61.29	9.11	10.00	9.92	12.54
		6	54.53	73.91	11.33	13.33	8.30	13.15
		8	50.11	61.90	14.08	17.46	8.84	12.34
	25	2	25.00	25.00	0.00	0.00	13.32	16.46
		4	45.42	61.29	9.08	9.68	10.39	13.19
		6	50.09	73.91	12.46	13.04	8.72	10.89
		8	51.83	60.00	13.06	14.75	9.11	11.93
	30	2	25.00	25.00	0.00	0.00	12.44	15.60
		4	41.42	61.29	8.77	9.68	8.33	11.20
		6	54.71	73.91	11.51	13.04	8.17	10.12
		8	51.96	80.00	13.17	16.13	7.02	9.19
	35	2	25.00	25.00	0.00	0.00	12.33	14.15
		4	38.06	61.29	9.68	9.68	9.36	11.94
		6	52.58	73.33	12.08	13.04	8.59	10.11
		8	50.13	80.33	13.61	14.75	7.73	9.68
	40	2	25.00	25.00	0.00	0.00	13.69	15.37
		4	45.03	61.29	8.47	9.68	10.04	12.54
		6	54.78	73.33	11.30	13.04	8.96	10.70
		8	61.66	79.66	12.45	14.75	8.14	11.21
	45	2	25.00	25.00	0.00	0.00	13.44	15.19
		4	45.42	61.29	9.08	9.68	10.16	12.61
		6	60.23	73.33	11.50	13.04	8.37	10.72
		8	65.33	78.64	12.56	14.47	6.25	8.60
	50	2	25.00	25.00	0.00	0.00	12.47	15.84
		4	53.42	61.29	9.71	10.00	9.82	11.29
		6	58.20	74.47	13.04	14.89	9.08	10.33
		8	56.39	61.29	14.35	16.13	7.93	9.60
	55	2	14.29	14.29	0.00	0.00	25.38	26.98
		4	55.56	55.56	3.70	3.70	22.90	24.27
		6	56.50	70.00	11.00	12.50	6.66	21.57
		8	65.33	73.33	11.11	11.11	8.08	9.55

Table 4.13 The lower bound performances, C3

TC	N	K	%DEV LB_1		%DEV LB_2		%DEV LB_3	
			Average	Maximum	Average	Maximum	Average	Maximum
1	20	2	33.33	33.33	0.00	0.00	20.56	22.74
		4	63.94	66.67	9.85	16.67	14.90	22.05
		6	59.81	76.47	20.01	25.00	12.34	17.84
		8	67.22	80.95	24.08	28.57	9.80	13.94
	25	2	33.33	33.33	0.00	0.00	19.72	21.99
		4	64.24	66.67	10.61	16.67	14.64	21.99
		6	55.29	75.00	23.01	25.00	10.50	15.24
		8	63.22	80.95	23.13	27.27	9.53	15.24
	30	2	33.33	33.33	0.00	0.00	19.36	20.71
		4	64.24	66.67	10.61	16.67	13.61	19.19
		6	57.17	75.00	18.25	25.00	10.33	12.61
		8	67.05	80.95	22.67	23.81	8.31	10.12
	35	2	33.33	33.33	0.00	0.00	18.88	21.20
		4	63.94	66.67	9.85	16.67	12.42	19.91
		6	57.50	75.00	20.00	25.00	10.23	13.54
		8	66.06	80.95	24.50	27.27	9.53	12.81
	40	2	33.33	33.33	0.00	0.00	19.53	21.31
		4	63.64	63.64	9.09	9.09	12.72	14.22
		6	57.50	75.00	22.50	25.00	10.45	13.11
		8	67.14	80.95	22.67	23.81	8.42	10.14
	45	2	33.33	33.33	0.00	0.00	19.76	20.52
		4	63.64	63.64	9.09	9.09	12.76	14.01
		6	62.50	75.00	21.25	25.00	10.38	12.32
		8	71.14	80.95	23.05	23.81	8.64	10.63
	50	2	33.33	33.33	0.00	0.00	19.37	21.73
		4	64.24	66.67	10.61	16.67	13.93	18.50
		6	60.29	76.47	19.78	25.00	10.98	15.44
		8	70.56	81.82	26.71	28.57	11.48	13.71
	55	2	50.00	50.00	0.00	0.00	18.91	20.60
		4	72.76	73.33	18.29	20.00	11.90	15.30
		6	75.32	81.82	24.16	27.27	7.93	9.55
		8	75.55	85.71	28.02	31.03	7.33	11.54
2	20	2	14.29	14.29	0.00	0.00	25.48	29.63
		4	55.71	57.14	4.05	7.14	23.76	27.08
		6	45.15	70.73	11.43	30.00	13.99	24.31
		8	54.53	73.33	11.36	13.33	8.86	12.43
	25	2	14.29	14.29	0.00	0.00	25.93	28.39
		4	55.87	57.14	4.39	7.14	24.42	27.43
		6	34.97	70.73	11.46	12.50	7.47	23.49
		8	49.09	73.33	10.93	13.04	10.09	11.70
	30	2	14.29	14.29	0.00	0.00	24.98	27.66
		4	55.87	57.14	4.39	7.14	22.88	24.98
		6	42.01	70.00	8.47	12.50	12.96	21.67
		8	54.42	73.33	10.70	13.04	8.08	10.12
	35	2	14.29	14.29	0.00	0.00	24.91	26.41
		4	55.71	57.14	4.05	7.14	22.57	25.12
		6	42.25	62.50	9.50	12.50	10.95	21.54
		8	52.46	73.33	11.88	13.04	8.77	10.51
	40	2	14.29	14.29	0.00	0.00	26.17	27.57
		4	55.56	55.56	3.70	3.70	23.88	25.69
		6	39.25	62.50	11.00	12.50	8.09	21.83
		8	54.48	73.33	10.70	13.04	9.42	10.70
	45	2	14.29	14.29	0.00	0.00	25.89	27.31
		4	55.56	55.56	3.70	3.70	23.41	24.75
		6	47.50	70.00	10.25	12.50	9.35	21.99
		8	59.82	73.33	10.71	11.11	8.61	10.37
	50	2	14.29	14.29	0.00	0.00	25.09	28.04
		4	55.87	57.14	4.39	7.14	23.23	25.93
		6	45.40	70.73	9.21	12.50	12.61	22.71
		8	58.14	74.47	13.06	14.89	9.29	11.11
	55	2	25.00	25.00	0.00	0.00	12.84	14.81
		4	60.90	61.29	8.77	9.68	9.37	12.06
		6	65.45	73.91	11.50	13.04	7.61	9.41
		8	65.90	80.00	13.04	13.33	7.43	9.13

Recall that BAB_2 and BAB_3 perform better than BAB_1. It can be observed from Tables 4.11, 4.12 and 4.13 that the average and maximum deviations of LB_2 and LB_3 are very low compared to deviations of LB_1. This verifies that the performance of the branch and bound algorithms is very much influenced from the quality of the lower bounds.

We observe from the tables that LB_2 and LB_3 do not consistently outperform each other. The average and maximum deviations of LB_2 decrease as the equipment cost variability, i.e., difference between equipment costs, decreases. This is due to the fact that the cheapest equipment cost is used in calculating LB_2. Note that, LB_2 finds the optimal solution at root node for K=2 when the cycle time is larger (*C2* and *C3*).

Recall that LB_3 is designed when there is no limit on the number of workstations. So, we expect that LB_3 works better in instances with higher number of workstations. It can be observed from the tables that the average and maximum deviations of LB_3 decrease as the number of workstations increases.

To set the limit of our branch and bound algorithm in terms of the number of tasks and number of workstations we design a small experiment with 60, 70 and 80 tasks problem instances for *C3*. The results are tabulated in Table 4.16. Note that *C3* is relatively easier than *C1* and *C2*. The results of our main experiment indicate that these problem sizes could not be solved in two hours with *C1* and *C2*.

Table 4.14 The performance of our branch and bound algorithm for large-sized problems, C3

TC	N	K	BAB_{23}						
			Average # of nodes	Maximum # of nodes	Average # of nodes till optimality	Maximum # of nodes till optimality	Average CPU time	Maximum CPU time	# of unsolved instances
1	60	2	51,160	51,162	110	119	0.052	0.062	0
		4	38,032	49,711	6,283	19,801	0.055	0.079	0
		6	579,929	868,846	19,841	36,123	0.950	1.828	0
		8	25,950,917	69,475,888	511,642	3,303,529	46.913	109.422	0
	70	2	3,143,100	3,143,106	130	139	2.966	2.969	0
		4	2,470,855	3,621,008	400,377	1,506,692	3.567	6.532	0
		6	158,830,518	408,046,375	47,641,085	232,697,893	231.663	625.265	0
		8	1,195,978,548	4,178,395,055	220,114,509	1,472,611,086	-	-	-
	80	2	12,098,700	12,098,707	150	159	39.997	40.062	0
		4	32,628,382	43,228,566	21,562,873	38,449,919	115.231	150.469	0
		6	1,946,247,758	2,018,767,841	161,306,402	1,612,925,079	-	-	-
2	60	2	51160	51,162	110	119	0	0.062	0
		4	38,032	49,711	6,283	19,801	0.052	0.078	0
		6	367,831	2,064,664	32,987	94,907	0.475	2.672	0
		8	1,446,850	4,893,711	413,938	3,269,869	2.356	9.547	0
	70	2	3,143,100	3,143,106	130	139	2.967	2.969	0
		4	2,470,855	3,621,008	400,377	1,506,692	3.566	6.516	0
		6	100,805,770	375,881,273	23,010,840	114,880,705	148.942	589.828	0
		8	423,122,289	1,544,933,516	194,024,013	1,177,726,182	627.524	2413.953	0
	80	2	12,098,700	12,098,707	150	159	39.981	40.000	0
		4	32,902,363	43,228,566	21,812,218	40,943,361	116.319	158.140	0
		6	1,081,403,807	2,129,167,830	13,375,451	24,005,720	3661.496	7200.000	3

As can be observed from the table for $N = 60$ all the instances are solved in two hours. When $N = 70$, for Set I up to 6 workstations are solved whereas for Set II up to 8 workstations are solved. For $N = 80$, when Set I is used, the instances up to 4 workstations are solved whereas when Set II is used and K is set to 6, 3 out of 10 instances remain unsolved after two hours.

CHAPTER 5

CONCLUSIONS

In this study, we develop an exact algorithm for a Flexible Assembly Line Design problem with fixed number of workstations. We assume the task times and equipment costs are correlated in the sense that the cheaper equipment gives no smaller task times. Given the cycle time we find the assignment of tasks and equipments to the workstations with minimum total equipment cost. In doing so, we develop a branch and bound algorithm that uses powerful lower bounds and reduction mechanisms. We also study a special case of the problem with identical task times and discuss the way we benefit from this case in developing a lower bound.

We design an experiment to test the performance of our branch and bound algorithm together with the reduction and bounding mechanisms. The number of tasks and number of workstations are the main factors that affect the difficulty of the problem. Generally, as the number of tasks and number of workstations increase, the solution times increase. We also observe that as the cycle time increases, the complexity of the solutions decreases. The results of our computational experiments reveal that in our termination limit of two hours up to 80 tasks with 6 workstations and 70 tasks with 8 workstations are solved when cycle time is large. When medium cycle time is used, up to 55 tasks and 6 workstations are solved, when cycle time is small up to 50 tasks and 8 workstations can be solved.

Other important factors that affect problem difficulty are the equipment costs and the flexibility ratio of the precedence relations. As the variability between the equipment costs decreases, the solution times decrease considerably. We observe the most difficult combination when the cycle time is low and the variability between equipment costs is high. Moreover we find that as an increase in the flexibility ratio adds to the difficulty of the solutions.

In general, the average and maximum number of nodes till optimality is very low compared to the number of nodes searched, in particular when K and N are large. Hence, one can use our branch and bound algorithm as a truncated approach if the guarantee of optimality is not as essential. In most of the instances the average performances are close to the maximums indicating the consistent behavior of our branch and bound algorithm. Moreover, our lower bounding procedures and reduction mechanisms are found to be very effective in reducing the size of the search.

To the best of our knowledge our study is the first attempt to solve the Flexible Line Design problem with fixed number of workstations. We hope our study helps to open new research areas most noteworthy of which are discussed below:

- In this study, we considered correlated task times and equipment costs. General task times and equipment costs may be a worth-studying extension.

- We used deterministic task times. Using stochastic task times may be considered as a future study.

- We assume all tasks are performed by a single equipment. Multiple equipments (tools) per task case can be a challenging research area.

- We take the cycle time as a constraint. The cycle time may be treated as a decision variable.

- Heuristic procedures –using our reduction and bounding mechanisms-- may be developed to solve larger sized problem instances.

- We assume all equipments can perform all tasks, a reasonable extension may be to assume each equipment is eligible to perform only a subset of tasks.

REFERENCES

Baybars, I., 1986. ‘A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem’, Management Science, 32, 8, 909-932.

Bukchin, J. and Tzur, M., 2000. ‘Design of Flexible Assembly Line to Minimize Equipment Cost’, IIE Transactions, 32, 585-598.

Bukchin, J.; Rubinovitz, J. 2003. ‘A Weighted Approach for Assembly Line Design with Station Paralleling and Equipment Selection’, IIE Transactions 35, 73 - 85.

Hackman, S.T., Magazine, M.J. and Wee, T.S., 1989. ‘Fast, Effective Algorithms for Simple Assembly Line Balancing Problems’, Operations Research, 37, 916-924.

Liu S.B., Ong H.L., Huang H.C., 2005. ‘A Bi-directional Heuristic for Stochastic Assembly Line Balancing Problems’ Int J Adv Manuf Technol, 25, 71-77.

Pekin, N. Azizoğlu, M., 2008, ‘Bi-criteria Flexible Assembly Line Design Problem with Equipment Decisions’, International Journal of Production Research, 46, 6323 – 6343.

Rekiek B., De Lit P., Pellichero F., Falkenauer E.,. Delchambre A., 1999a. 'Applying the Equal. Piles Problem to Balance Assembly Lines', Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning.

Rubinovitz, J. and Bukchin, J., 1993. ‘RALB - A Heuristic Algorithm for Design and Balancing of Robotic Assembly Lines’, Annals of the CIRP, 42, 497-500.

Scholl A., 1993, ‘Data of Assembly Line Balancing Problems’, http://www.assembly-line-balancing.de/

Ugurdag, H.F., Rachamadugu, R., Papachristou, C.A., 1997. “Designing Paced Assembly Lines with Fixed Number of Stations”, European Journal of Operational Research,102, 488-501.

APPENDIX A

Table A.1 The maximum number of nodes and CPU times of the branch and bound algorithm, Set I

CT	N	K	BAB_1		BAB_2		BAB_3		BAB_{12}		BAB_{23}		BAB_0	
			Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time
1	20	2	206	0.016	206	0.000	206	0.000	206	0.000	206	0.000	206	0.000
		4	3,884	0.016	2,816	0.016	3,880	0.015	2,816	0.016	2,816	0.016	3,884	0.016
		6	23,265	0.032	17,320	0.016	20,146	0.031	17,320	0.031	17,052	0.031	23,265	0.032
		8	116,853	0.156	80,716	0.109	71,092	0.094	80,716	0.125	63,774	0.078	116,853	0.125
	25	2	449	0.000	449	0.000	449	0.000	449	0.000	449	0.000	449	0.016
		4	12,080	0.016	10,615	0.016	12,080	0.016	10,615	0.016	10,615	0.016	12,080	0.016
		6	198,140	0.187	155,825	0.157	111,925	0.125	155,651	0.171	99,060	0.110	198,329	0.157
		8	988,360	1.156	631,780	0.781	337,250	0.406	620,305	0.781	231,745	0.312	1,044,087	1.000
	30	2	7,829	0.016	7,829	0.016	7,829	0.016	7,829	0.016	7,829	0.016	7,829	0.016
		4	3,239,886	3.328	2,717,212	2.859	3,237,614	3.313	2,717,212	2.860	2,717,212	2.859	3,239,886	3.157
		6	120,078,338	145.344	85,216,503	108.719	56,876,108	77.578	85,216,503	109.516	51,471,522	69.719	120,078,338	127.438
		8	3,037,907,082	4049.016	1,509,342,461	2227.187	343,188,357	565.250	1,509,342,461	2248.531	324,677,324	526.157	3,037,907,422	3461.782
2	20	2	265	0.000	265	0.000	265	0.000	265	0.000	265	0.000	265	0.000
		4	1,236	0.000	1,160	0.016	1,236	0.015	1,160	0.000	1,160	0.016	1,236	0.000
		6	8,536	0.016	4,414	0.016	8,376	0.016	4,414	0.016	4,414	0.016	8,536	0.016
		8	25,692	0.032	16,462	0.031	20,656	0.031	16,462	0.031	15,216	0.016	25,692	0.016
	25	2	595	0.016	595	0.000	595	0.015	595	0.016	595	0.016	595	0.000
		4	2,286	0.015	2,254	0.016	2,282	0.016	2,254	0.016	2,254	0.016	2,286	0.016
		6	27,856	0.031	16,106	0.016	25,114	0.031	16,106	0.016	15,802	0.016	27,856	0.016
		8	141,127	0.156	62,253	0.078	73,209	0.078	62,253	0.078	51,733	0.063	142,153	0.125
	30	2	7,984	0.016	7,984	0.016	7,984	0.016	7,984	0.016	7,984	0.016	7,984	0.016
		4	98,468	0.094	90,950	0.093	98,468	0.094	90,950	0.093	90,950	0.094	98,468	0.094
		6	3,526,989	4.328	1,470,864	2.109	3,191,570	4.063	1,470,864	2.094	1,469,142	2.094	3,526,989	3.891
		8	61,011,706	78.718	32,641,829	48.422	34,477,867	49.360	32,641,829	48.438	27,119,290	39.984	61,011,706	68.719

Table A.2 The maximum number of nodes and CPU times of the branch and bound algorithm, Set II

CT	N	K	BAB_1		BAB_2		BAB_3		BAB_{12}		BAB_{23}		BAB_0	
			Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time
1	20	2	206	0.000	206	0.000	206	0.000	206	0.000	206	0.000	206	0.000
		4	1,922	0.015	1,918	0.000	1,918	0.016	1,918	0.016	1,918	0.015	1,922	0.016
		6	13,366	0.016	11,942	0.016	12,071	0.016	11,942	0.016	10,597	0.016	13,366	0.016
		8	29,586	0.047	25,851	0.032	19,662	0.031	25,851	0.047	17,598	0.031	29,586	0.032
	25	2	449	0.000	449	0.000	449	0.000	449	0.015	449	0.000	449	0.016
		4	10,231	0.016	9,545	0.016	10,231	0.016	9,545	0.016	9,545	0.015	10,231	0.016
		6	82,754	0.094	78,837	0.078	77,782	0.079	78,663	0.078	73,703	0.078	82,928	0.078
		8	222,504	0.250	178,017	0.250	211,841	0.250	173,421	0.234	158,033	0.203	222,504	0.204
	30	2	7,829	0.016	7,829	0.016	7,829	0.016	7,829	0.016	7,829	0.016	7,829	0.016
		4	239,656	0.266	234,400	0.265	234,400	0.281	234,400	0.281	234,400	0.266	239,656	0.265
		6	38,020,862	39.500	33,161,757	34.766	23,666,230	26.344	33,159,837	35.109	20,307,181	23.218	38,022,782	36.750
		8	222,487,606	352.594	177,229,728	286.062	137,899,093	214.484	177,229,728	290.484	99,385,259	161.578	222,487,606	296.797
2	20	2	206	0.000	206	0.000	206	0.000	206	0.000	206	0.000	206	0.000
		4	690	0.015	686	0.016	690	0.015	686	0.016	686	0.000	690	0.000
		6	3,050	0.016	3,012	0.016	3,046	0.016	3,012	0.000	3,012	0.016	3,050	0.015
		8	16,518	0.016	13,034	0.016	13,490	0.016	13,034	0.016	10,098	0.016	16,518	0.016
	25	2	485	0.000	485	0.016	485	0.015	485	0.000	485	0.000	485	0.000
		4	2,260	0.000	2,254	0.015	2,260	0.016	2,254	0.015	2,254	0.015	2,260	0.000
		6	13,887	0.016	12,927	0.016	13,729	0.016	12,927	0.016	12,769	0.016	13,887	0.016
		8	41,604	0.047	36,450	0.047	41,604	0.047	36,450	0.032	36,450	0.032	41,604	0.047
	30	2	6,932	0.016	6,932	0.016	6,932	0.016	6,932	0.016	6,932	0.016	6,932	0.016
		4	7,773	0.016	7,767	0.016	7,773	0.016	7,767	0.016	7,767	0.016	7,773	0.016
		6	392,035	0.438	389,919	0.438	381,761	0.422	389,919	0.438	379,645	0.422	392,035	0.422
		8	4,623,325	6.250	4,529,413	6.140	4,119,541	5.500	4,529,413	6.140	4,025,629	5.406	4,623,325	5.485

Table A.3 The worst case performances with and without elimination rules

TC	CT	N	K	BAB_2				BAB_{23}			
				With Reductions		Without Reductions		With Reductions		Without Reductions	
				Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time	Maximum # of nodes	Maximum CPU time
1	1	20	2	206	0.016	387	0.000	206	0.000	387	0.000
			4	3,884	0.016	5,730	0.015	2,816	0.016	5,654	0.015
			6	23,265	0.032	113,329	0.125	17,052	0.031	52,654	0.078
			8	116,853	0.156	522,303	0.578	63,774	0.078	155,266	0.187
		25	2	449	0.000	1,322	0.015	449	0.000	1,322	0.000
			4	12,080	0.016	23,706	0.031	10,615	0.016	20,677	0.032
			6	198,140	0.187	511,390	0.500	99,060	0.110	323,232	0.344
			8	988,360	1.156	5,287,023	5.610	231,745	0.312	2,462,116	2.735
		30	2	7,829	0.016	8,650	0.016	7,829	0.016	8,650	0.016
			4	3,239,886	3.328	8,761,500	9.187	2,717,212	2.859	8,717,258	9.157
			6	120,078,338	145.344	515,772,243	571.610	51,471,522	69.719	188,267,827	226.781
			8	3,037,907,082	4049.016	4,022,593,885	-(5 unsolved)	324,677,324	526.157	872,298,032	1262.859
	2	20	2	265	0.000	849	0.000	265	0.000	849	0.000
			4	1,236	0.000	2,193	0.016	1,160	0.016	2,193	0.016
			6	8,536	0.016	8,546	0.016	4,414	0.016	8,227	0.016
			8	25,692	0.032	54,496	0.062	15,216	0.016	44,918	0.047
		25	2	595	0.016	1,757	0.015	595	0.016	1,757	0.015
			4	2,286	0.015	5,913	0.016	2,254	0.016	5,913	0.016
			6	27,856	0.031	83,235	0.094	15,802	0.016	76,162	0.094
			8	141,127	0.156	347,469	0.391	51,733	0.063	248,321	0.297
		30	2	7,984	0.016	8,268	0.016	7,984	0.016	8,268	0.016
			4	98,468	0.094	177,719	0.235	90,950	0.094	177,719	0.234
			6	3,526,989	4.328	5,850,952	7.922	1,469,142	2.094	5,607,896	7.610
			8	61,011,706	78.718	178,869,762	241.125	27,119,290	39.984	128,874,074	174.265
2	**1**	**20**	2	206	0.000	387	0.000	206	0.000	387	0.016
			4	1,918	0.000	4,002	0.015	1,918	0.015	4,002	0.016
			6	11,942	0.016	25,075	0.032	10,597	0.016	23,203	0.032
			8	25,851	0.032	80,710	0.093	17,598	0.031	46,360	0.063
		25	2	449	0.000	1,322	0.016	449	0.000	1,322	0.000
			4	9,545	0.016	20,374	0.031	9,545	0.015	20,374	0.032
			6	78,837	0.078	247,192	0.266	73,703	0.078	166,431	0.204
			8	178,017	0.250	1,423,529	1.609	158,033	0.203	809,912	0.938
		30	2	7,829	0.016	8,650	0.016	7,829	0.016	8,650	0.016
			4	234,400	0.265	563,248	0.781	234,400	0.266	561,984	0.766
			6	33,161,757	34.766	102,267,730	137.875	20,307,181	23.218	46,633,390	64.266
			8	177,229,728	286.062	1,256,143,870	1790.797	99,385,259	161.578	395,404,570	579.313
	2	**20**	2	206	0.000	849	0.000	206	0.000	849	0.016
			4	686	0.016	1,970	0.016	686	0.000	1,970	0.016
			6	3,012	0.016	6,011	0.015	3,012	0.016	6,011	0.016
			8	13,034	0.016	20,522	0.031	10,098	0.016	19,930	0.031
		25	2	485	0.016	1,757	0.000	485	0.000	1,757	0.015
			4	2,254	0.015	4,964	0.016	2,254	0.015	4,964	0.016
			6	12,927	0.016	51,637	0.063	12,769	0.016	50,817	0.063
			8	36,450	0.047	156,815	0.187	36,450	0.032	142,661	0.156
		30	2	6,932	0.016	8,268	0.016	6,932	0.016	8,268	0.016
			4	7,767	0.016	10,154	0.016	7,767	0.016	10,154	0.016
			6	389,919	0.438	739,690	1.032	379,645	0.422	739,690	1.016
			8	4,529,413	6.140	11,988,561	16.906	4,025,629	5.406	9,129,805	12.625