DISASSEMBLY LINE BALANCING PROBLEM WITH FIXED NUMBER OF WORKSTATIONS AND FINITE SUPPLY

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

BY

EDA GÖKSOY

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

JUNE 2010

Approval of the thesis:

DISASSEMBLY LINE BALANCING PROBLEM WITH FIXED NUMBER OF WORKSTATIONS AND FINITE SUPPLY

submitted by **EDA GÖKSOY** in partial fulfillment of the requirement 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**	________________
Prof. Dr. Sencer Yeralan Co-Supervisor,**Agricultural and Biological Engineering Dept., University of Florida, USA**	________________

Examining Committee Members

Assoc. Prof. Dr. Tayyar Şen Industrial Engineering Dept., METU	________________
Prof. Dr. Meral Azizoğlu Industrial Engineering Dept., METU	________________
Prof. Dr. Sencer Yeralan Agricultural and Biological Engineering Dept., University of Florida, USA	________________
Asst. Prof. Dr. Cem İyigün Industrial Engineering Dept., METU	________________
Dr. Ünal KOYAZ ASELSAN A.Ş.	________________
Date:	**21.06.2010**

ABSTRACT

DISASSEMBLY LINE BALANCING PROBLEM WITH FIXED NUMBER OF WORKSTATIONS AND FINITE SUPPLY

Göksoy, Eda

M.S., Department of Industrial Engineering
Supervisor : Prof. Dr. Meral Azizoğlu
Co-Supervisor: Prof. Dr. Sencer Yeralan

June 2010, 76 pages

In this thesis, we consider a Disassembly Line Balancing Problem (DLBP) with fixed number of workstations. We aim to maximize the total value of the recovered parts.

We assume that there is a limited supply for the products to be disassembled. Different components can be obtained by disassembling different units of the product. Our aim is to assign the tasks to the workstations of the disassembly line so as to maximize the total value of the recovered parts. We present several upper and one lower bounding procedure. The results of our computational study have revealed the satisfactory behavior of our bounding mechanisms.

Keywords: Disassembly Process, Line Balancing, Linear Programming Relaxation

ÖZ

SABİT SAYIDA İSTASYON VE SONLU ARZ İÇEREN DEMONTAJ HAT DENGELEME PROBLEMİ

Göksoy, Eda

Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Meral Azizoğlu
Ortak Tez Yöneticisi : Prof. Dr. Sencer Yeralan

Haziran 2010, 76 Sayfa

Bu çalışmada, sabit sayıda istasyon içeren Demontaj Hattı Dengeleme Problemi ele alınmıştır. Amacımız, geri kazandırılan parçaların toplam değerini ençoklamaktır.

Demonte edilecek ürünlerin arzının sınırlı olduğunu varsaydık. Ürünün değişik birimlerinin demonte edilmesiyle değişik parçalar elde edilebilmektedir. Amacımız, işleri demontaj hattındaki istasyonlara, geri kazandırılan parçaların toplam değerini ençoklayacak şekilde, atamaktır. Birçok üst ve bir alt sınır prosedürleri sunduk. Deneysel sonuçlarımız sınırlama mekanizmalarımızın tatmin edici davrandığını göstermiştir.

Anahtar Kelimeler: Demontaj Süreci, Hat Dengeleme, Doğrusal Programlama Gevşetmesi

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisors, Prof. Dr. Meral Azizoğlu and Prof. Dr. Sencer Yeralan for their supervision, patience and understanding. Studying with them is not only instructive, but also a very pleasant experience. Without their support and encouragement, this study would be harder to complete and succeed.

I would like to thank jury members for their valuable contributions on the thesis.

I owe my deepest gratitude to my family members: Emine Göksoy Pektaş, Kamil Göksoy and Hacer Göksoy for their unconditional support and love.

I would like to thank my company, ASELSAN, for the support about thesis study permissions. I would like to thank my manager Ünal Koyaz and Arzu Konukseven for allowing me to go for thesis study whenever I needed and Leyla Demirel for her support and covering my vacancy.

This thesis could not be realized without the support of Murat Kalaycılar who enriched my life and made me smile during every single step of this study. I thank with all my heart to him and there is not a way to express my appreciation.

I would like to thank my dear friends Yasin Gökpınar, Özgün Töreyen, Başak Dilber, Oğuz Yağcı, Esra Eroğlu and Çağla Ateşalp for their cheerful company and invaluable frienship; Çiğdem Cihangir for her cheer and being an excellent homemate; Diclehan Tezcaner, Gülşah Karakaya, Murat Aslan and Özgür Gürses for their insight, wisdom and invaluable support.

I would like to thank TÜBİTAK for the funding they have provided during my M.S. study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In recent years, the term “sustainability” has become quite a buzzword. Its meaning has been somewhat ambiguous, or perhaps, dependent on the context. The recent concerns regarding such global catastrophic phenomena as climate change or wide ranging financial insolvencies have given impetus to the “green push.” In some sense, sustainability is recognized to have three components: environmental, economic and social. However, from an engineering perspective, there seems to be two important aspects of sustainability. First, there is good reason to develop new disruptive technologies that would enable society to reduce its environmental footprint. This means that the same products and services may be provided with fewer demands on the environment. Such a reduction on the use of environmental resources is sought through advanced technologies. However, until such disruptive technologies take hold, one also considers conservation as an interim path to environmental footprint reduction. These measures usually revolve around reuse and recycling. Our study here is in line with conservation. Not only do we advocate reuse, but also facilitate the operationally most productive and profitable way to reuse. The latter emphasis is due to the recognition that industry will more easily embrace greener practices if it also provides reasonable rates of financial returns. In short, our work bodes well with contemporary urges to become greener, while maintaining the industrial sensitivities to financial concerns.

The importance of environmental issues has been recognized by many manufacturers and product designers. This recognition is triggered by the new and more rigid environmental regulations and increased customer awareness towards environmental issues. Moreover the recent advances in technology have made it possible to manufacture the products that meet the environmental standards and are easy to reuse after consumed by the customer.

The environmental regulations, customer awareness and recent advances in technology all together have shifted the product recovery process from the act of disposing to the act of reusing and recycling.

Recycling preserves the material content of the discarded (used) products via some manufacturing and disassembly operations. Remanufacturing, on the other hand, keeps the functional content of the used products and improves their quality up to a desired usable level via some manufacturing and disassembly operations.

Disassembly operations involve the separation of the reusable parts from the discarded products. Those parts are either subject to remanufacturing operations or sold to suppliers.

The disassembly operations are usually performed on a disassembly line that consists of a number of serial workstations. The first workstation takes the product to be disassembled and the parts are disconnected on different workstations. A cycle terminates, that is the product leaves the line, whenever all its required parts are disassembled.

A disassembly line balancing problem finds the set of tasks assigned to each workstation for each product to be disassembled. The problem is critical in minimizing the use of valuable resources (such as time and money) invested in disassembly, and maximizing the level of automation of the disassembly process and the quality of the parts or materials recovered (McGovern and Gupta, 2007).

In this thesis, we consider a Disassembly Line Balancing Problem (DLBP) with a fixed number of workstations so as to maximize the value of recovered parts. We assume there is a limited supply for the products to be disassembled. Different components can be obtained by disassembling different units of the product. Our aim is to assign the tasks to the workstations of the disassembly line so as to maximize the total value of the recovered parts. We assume each part has a unit profit defined and a specified demand, hence it may require many products to be disassembled.

To the best of our knowledge, our study is the first attempt to tackle DLBP with profit maximization and with a finite supply and a fixed number of workstations. The rest of the thesis is organized as follows: In Chapter 2, we review the disassembly process and literature on disassembly lines. We define our problem in Chapter 3. The chapter includes two alternate mathematical formulations of the disassembly line balancing problem and settles the complexity of the problem. In Chapter 4, we discuss the linear programming (LP) relaxation together with the mechanisms to strengthen it. We present LP-based heuristic procedures in Chapter 5. Our computational experiment is discussed in Chapter 6. We conclude in Chapter 7, by stating our main findings and pointing out future research directions.

CHAPTER 2

THE DISASSEMBLY PROCESS AND RELATED LITERATURE

In this chapter, we first define disassembly process, and then discuss assembly line balancing and disassembly line balancing problems. Finally we review the previous studies on the Disassembly Line Balancing Problems.

2.1. Disassembly Process

Güngör and Gupta (2001) define *disassembly* as a systematic process of separating a product into its constituent parts, components, subassemblies or other groupings.

As mentioned by Brennan et al.(1994), disassembly covers both economic and environmental concerns such as *discontinued products* (leading excess inventory of undesirable assemblies), *reduction in lead time* (as disassembled products can satisfy some scarce products or some that are urgent in demand), *forced disassembly* (disassembly is imposed by governments due to recycling regulations).

Lambert (2002) emphasizes that disassembly process does not imply reverse assembly process.

In general, disassembly process is divided into two as *partial disassembly* and *complete disassembly*. In the first one, the product is not fully disassembled while the product is fully disassembled in the latter. Incomplete disassembly can be favored en route to minimizing excess materials by considering both economic and environmental aspects.

Operations planning issues in assembly environments are much well known than the disassembly environments as disassembly systems are recognized later than the assembly systems. Brennan et al. (1994) state that even the assembly and disassembly systems have similarities, disassembly causes many problems in operations management. Some operational impacts of incorporating disassembly are as discussed below:

- Impact on Product Cost: Labor, energy and overheads might be incurred more during disassembly rising the product costs.
- Impact on Financial Decisions: Longer planning horizon is needed due to the increased uncertainties of disassembly process, thus capital budgeting process might be more difficult.
- Impact on Capacity and Storage Requirements: Due to the uncertainties in product life cycles, forecasting the demand of disassembled products might be hard and lead to variations. Variation in demand forecasts might increase capacity and storage requirements.

2.2. Assembly Lines versus Disassembly Lines

Assembly lines are special flow-line production systems which are typical in the industrial production of high quantity standardized commodities. In the literature there exist several classification schemes for the assembly lines. According to the nature of the products, operation modes and nature of operation times, the following classifications are made:

- Nature of Products
    - ✓ Single-Model Lines
    - ✓ Mixed-Model Lines
    - ✓ Multi-Model Lines

- Operation Mode
    - ✓ Paced Lines (Transfers between the workstations are synchronous)
    - ✓ Unpaced Lines (Transfers between the workstations are not synchronized)
- Nature of Operation Times
    - ✓ Deterministic Operation Times (Known certainly)
    - ✓ Stochastic Operation Times (Known by probability function)
    - ✓ Dynamic Operation Times (Subject to change)
    - ✓ Static Operation Times (Not subject to change)

There are two types of assembly line balancing problems: Type I and Type II.

- Type I problems assume a fixed cycle time while minimizing number of stations.
- Type II problems assume a fixed and given stations while minimizing the cycle time.

For the details of the assembly lines and assembly line balancing problems, one may refer to the textbook by Scholl (1999) and a review paper by Baybars (1986).

To have a better understanding of the disassembly lines, we discuss the similarities and differences between assembly lines and disassembly lines.
Brennan et al. (1994) study the assembly lines and disassembly lines both from technical and operational points of view and present a comparison as shown in Table 2.1.

Table 2.1: The Comparison of the operational and technical considerations of the assembly and disassembly lines.

Line Considerations	**Assembly Line**	**Disassembly Line**
Demand	Dependent	Dependent
Demand Sources	Single	Multiple
Demanded Entity	End Product	Individual parts/subassemblies
Precedence relationships	Yes	Yes
Complexity related to precedence relationships	High(includes physical and functional precedence constraints)	Moderate (mostly physical constraints)
Uncertainty related to quality of parts	Low	High
Uncertainty related to quantity of parts	Low	High
Uncertainty related to WSs and the material handling system	Low to moderate	High
Reliability of the WSs and the material handling system	High	Low
Multiple products	Yes	Yes
Flow process	Convergent	Divergent
Line flexibility	Low to moderate	High
Layout alternatives	Multiple	Multiple
Complexity of performance measures	Moderate	High
Known performance measures	Numerous	N/A
Required line robustness	Moderate	High
Complexity of "between workstation inventory" handling	Moderate	High
Known techniques for line optimization	Numerous	None
Problem complexity	NP-hard	NP-hard

2.3. Literature on Disassembly Line Balancing Problems

Güngör and Gupta (2002)'s study introduces the disassembly line balancing problem. They discuss the importance of disassembly lines in product recovery, various complications to create an efficient disassembly line. Some considerations discussed in the study are stated as below:

- *Product Considerations*: Characteristic of a disassembly line depends on the products' variety disassembled on the same line. Disassembly line may deal with only one type of product or may disassemble a product family. The line may also receive several types of products.
- *Line Considerations:* Layouts inspired from the assembly lines and line speed are the most important considerations. The lines may be configured as serial, parallel, circular, U-shaped, cellular and two-sided lines. As operation mode, paced and unpaced transfers can be used.
- *Part Considerations:* There exists a serious uncertainty in the quality of the products disassembled. Proper or improper usage of the parts determines the defective or non-defective parts. The quantity of products disassembled differs according to their upgrading or downgrading during their usage.
- *Operational Considerations:* Disassembly task times may be considered as deterministic, stochastic and dynamic. These task times may vary depending on several factors that are related on the condition of the product and state of the disassembly workstation (worker). Furthermore, tasks may behave differently in environments having product-defects. The actions that are taken under such situations include; leaving the workstation early, skipping to the next workstation(s), disappearing from the line, revisiting to a preceding workstation or splitting into two or more (exploding).
- *Demand Considerations:* Three demand types may exist: The demand may be for one part only (single-part disassembly), or for multiple parts (partial disassembly) or demand for all parts (complete disassembly).

- *Assignment Considerations:* There are some restrictions limiting the assignment of tasks to the workstations. Some of these restrictions include assigning to a specific workstation for minimizing distance traveled or grouping tasks requiring similar operating conditions, availability of special machining and tooling at certain workstations, minimizing the disassembly direction changes and number of tool changes.
- *Other Considerations:* There are additional uncertainty factors related with the reliability of the workstations. Some parts may cause pollution or nuisance increasing the chance of breakdowns or downtimes of workstations.

Altekin et al. (2008) study the Disassembly Line Balancing Problem (DLBP) under partial disassembly. Their objective is to maximize total profit obtained from disassembling a single product. Their assumptions are as follows:

- Single type of product is disassembled on a paced line.
- There is an infinite supply of used product.
- All parameters are deterministic and static.
- A disassembly task cannot be split among two or more stations.
- A disassembly task may result in removal of one or more parts.
- A station cost is incurred for opening and operating a station per unit time.

They formulate their problem as a mixed integer linear program. Their model determines; the parts whose demand is satisfied to generate revenue, the tasks that will release selected parts, the number of opened stations, cycle time and feasible assignment of selected tasks to the stations considering the precedence relations. Using the linear programming relaxation of this formulation, they find upper and lower bounds on the total profit value. The results of their computational analysis on ten basic problems show that their bounds provide near optimal solutions for small sized problems and are capable of handling larger sized problems with up to 320 disassembly tasks in reasonable time.

McGovern and Gupta (2007) solve the DLBP using exhaustive search and a combinatorial optimization methodology. A new method for quantifying the level of balancing is proposed. Their method minimizes the number of workstations while balancing the idle times between the workstations. Exhaustive search works well enough in obtaining optimal solutions for small sized instances; however its exponential time complexity limits its application on the large sized instances. They also develop a genetic algorithm that involves a randomly generated initial population with cross-over, mutation and fitness competition performed over many generations. The algorithm finds optimal or near-optimal solutions, for the large sized problem instances.

Güngör and Gupta (2002) study the DLBP under complete disassembly. They consider the utilization of the resources efficiently while satisfying the demand. Their concerns are finding the minimum number of the disassembly workstations and improving the layout and material handling features of the disassembly line. They propose a heuristic to solve the DLBP under the following assumptions:

- The disassembly line is paced.
- One type of product is disassembled and each product have identical configuration.
- The supply of products is infinite.
- The complete disassembly is considered.
- All parameters are deterministic and static.

They use the disassembly of a PC consisting of eight tasks and eight parts, to illustrate their priority-rule-based and station oriented heuristic.

Güngör and Gupta (2001) discuss the DLBP in the presence of task failures. They assume if a task fails, none of its successors can be performed and discuss the complications of such failures on the disassembly line. Their problem is to assign tasks to workstations such that the effect of the defective parts on the disassembly line is minimized. They state their basic assumptions as:

- One type of product is disassembled.

- All parameters are deterministic and static.
- Probabilities of parts being defective are constant and known.

They propose a 5-steps solution approach. Firstly, an Incomplete State Network (ISN) representing all feasible states and their partial relationships are generated. Then all possible relationships are developed among the states of ISN, which results in complete ISN called State Network (SN). There are edges in SN which implies relations between the states of SN. In the third step, the idle times and weights of task assignments for each edge are calculated to generate the Weighted State Network (WSN). In the fourth step shortest directed paths (SDP) between the source and the final nodes of WSN are found by the Dijkstra's shortest path algorithm. SDP is defined as a task assignment resulting in the minimum idle time on the line. In the final step, the cost of the complications for each alternative is calculated and the alternative with minimum complication cost is selected.

Our disassembly line balancing problem differs from the previous ones in the sense that we consider fixed number of workstations and finite supply of used products.

CHAPTER 3

PROBLEM DEFINITION

We consider a disassembly line balancing model so as to maximize the total net revenue. We assume the supply, i.e., the number of the product to be disassembled is known and each unit of the product will be disassembled in one period.

The tasks that release parts are referred to as part releasing tasks. All tasks have costs, the part releasing tasks, simply called parts, have revenues as well. Our problem is to assign the tasks to the workstations of the disassembly line for each disassembled product, i.e., in each period.

We make the following additional assumptions:

- The workstations are already mounted and there are *K* workstations.
- All workstations are equipped identically and can perform all tasks at the same pace.
- There is a single disassembly product with finite supply rate, *S.*
- All units of the product contain all disassembly parts with no differentiation.
- All parameters, i.e., task times, part demands, costs are known with certainty, i.e., deterministic.
- The parameters are not subject to any change, i.e., the system is static.
- The cycle time, i.e., the time can be allocated to each workstation, is deterministic and static.

- Each task is specified by its cost and processing time. The part releasing tasks have additional parameter, i.e., revenues.
- Each period is specified by a single unit disassembly, hence there are *S* periods.
- *S* units of the product are sufficient for all demand, hence no shortages are allowed. We also refer to each period as a cycle. Hence there are *S* cycles.
- The tasks are not indivisible, i.e., they should be assigned to exactly one workstation.
- There is a partial disassembly, and there can be different disassembled parts in different periods.

In Section 3.1 we explain the precedence relations that are used in disassembly systems, Section 3.2 defines our mathematical models. We set the complexity of the problem in Section 3.3. In Section 3.4 we provide an example problem to clarify our decisions.

3.1. Precedence Relations

There are basically 2 types of precedence relations in disassembly networks. These are *"AND"* type precedence relations and *"OR"* type precedence relations.

i. "AND" type precedence relations

"AND" type precedence relations between two tasks imply that one task can not start before the other finishes. The following figure illustrates 'AND' type precedence relations:

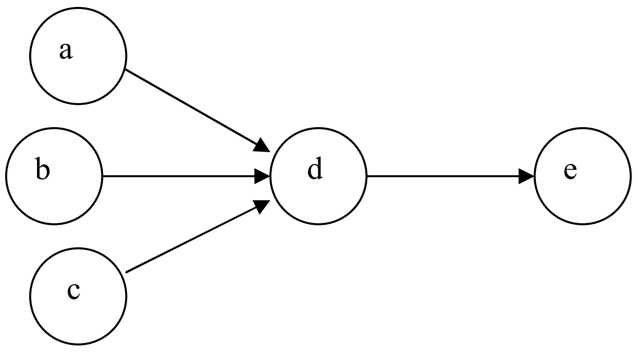


Figure 3.1: "AND" type precedence relations

According to the above figure, *tasks a, b, c* and *d* are predecessors of *task e*. These four tasks should be complete before *task e* starts. *Task d* is the immediate predecessor of *task e* as there is no task in between.

If *task i* is predecessor/immediate predecessor of *task j* then *task j* is successor/immediate successor of *task i*. Accordingly *task e* is successor of all other tasks and immediate successor of *task d*. *Task d* is the immediate successor of *tasks a, b and c.*

For example, in personal computer (PC) disassembly (Gupta, 2002) shown in Figure 3.5, both *tasks 2, 3, 5* and *6* should be performed to start *task 8*.

ii. "OR" type precedence relations

"OR" type relations are specific to the disassembly networks. There are two types of "OR" relations.

- **"OR" type predecessor precedence relations (POR)**

"POR" relations imply that at least one of the tasks in a specified set should be complete before another task begins. Figure 3.2 illustrates "POR" type precedence relations.

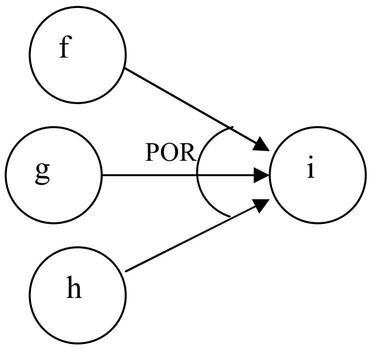


Figure 3.2: "POR" type precedence relations

According to the above figure, at least one of the tasks in set *{f, g, h}* should be complete before *task i* begins.

For example, in radio disassembly (Lambert, 1997) shown in Figure 5.5, either *Task 3* or *Task 4* should be disassembled to perform *Task 31*.

- **"OR" type successor precedence relations (SOR)**

"SOR" relations imply that at most one of the tasks in a specified set can be performed after one task completes. Figure 3 illustrates SOR type precedence relations.

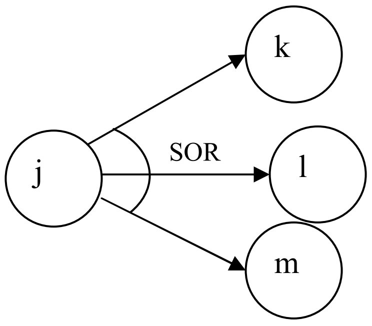


Figure 3.3: "SOR" type precedence relations

In figure 3, the tasks in set *{k, l, m}* are OR successors of *task j*. Accordingly after completing *task j* at most one of the tasks in the set can be performed.

For example in ball-point pen disassembly (Lambert, 1997) with "SOR" relationships, shown in Figure 3.4, either Task 2 or Task 3 can be performed after Task 1.

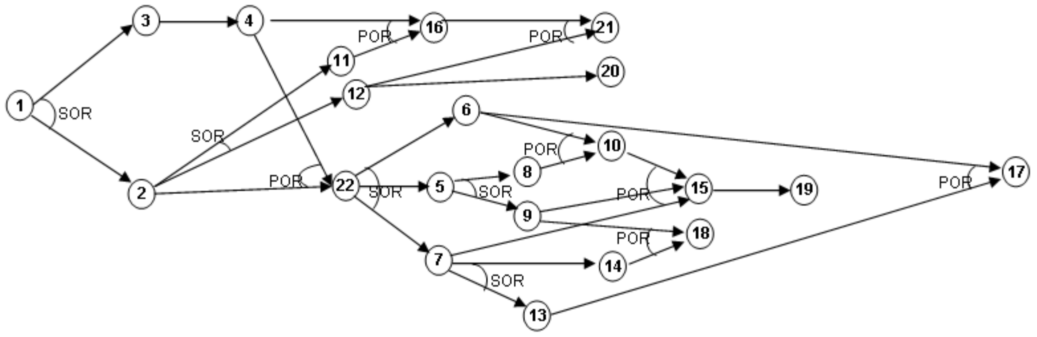


Figure 3.4: Precedence diagram of the Lambert's ball-point pen example (22 tasks) with "SOR" type precedence relations

In our model, we consider *"AND"* and *"POR"* type precedence relations. We assume our precedence network is deterministic and is not subject to any change (static).

3.2. Mathematical Model

In this section, we present our models that use the assumptions discussed and precedence network with *"AND"* and *"POR"* relations. We develop two models. Model I is a classical presentation whereas Model II is based on some optimality properties. Our initial experiments reveal that Model I is more efficient, hence used to find optimal solutions.

For both models we use the following parameters to define the problem size.

M: number of all tasks

N: number of part releasing tasks

K: number of disassembly products or periods

S: number of workstations

The task, period and workstation related indices are:

i: index of tasks , i=1,2,…,M

k index of workstations, k=1,2,…,K

t: index of periods, t=1,2,…,S

The following parameters and decision variables are valid for both models.

Parameters:

CT: maximum cycle time allowed for any of the workstations

t_i: processing time of task *i*

d_i: demand of part that is released by task *i*

K: maximum number of workstations that can be used

P_i: net revenue of task *i* ("revenue obtained by releasing a part"-"cost of task")

= r_i-c_i where r_i is the revenue due to task *i* and c_i is the cost of task *i* ($r_i = 0$ if task i does not release a part).

$PAND_i$: index set of AND predecessors of task *i*

POR_i : index set of OR predecessors of task *i*

Decision Variables:

Our decisions are assignments of tasks to workstations in each period. These assignments are explained by the following decision variables:

$$x_{ikt} = \begin{cases} 1, \text{if task i is assigned to workstation k in period t} \\ 0, \text{otherwise} \end{cases}$$

The following constraints are valid for both models. Model II uses some additional constraints.

Constraints:

The demand of the part releasing tasks should be satisfied.

$$\sum_{k=1}^{K}\sum_{t=1}^{s} x_{ikt} \geq d_i \qquad \forall i \qquad (c1)$$

The cycle time limit should not be exceeded.

$$\sum_{i=1}^{m} t_i x_{ikt} \le CT \qquad \forall k,t \qquad (c2)$$

The "AND" precedence relations should be satisfied. This constraint allows assignment of task i to station k if and only if all its "AND" predecessors are assigned to stations 1 through k.

$$\sum_{k=1}^{k} x_{ikt} \le \sum_{k=1}^{k} x_{lkt} \qquad \forall i,k,t,\ l \in PAND(i) \qquad (c3)$$

The "POR" precedence relations should be satisfied. This constraint allows the assignment of task i to station k if and only if *at least* one of "POR" predecessors of task i is assigned to stations 1 through k.

$$x_{ikt} \le \sum_{h=1}^{K} \sum_{b \in POR(i)} x_{bht} \qquad \forall i,k,t \qquad (c4)$$

A task can be assigned to at most one station in each period.

$$\sum_{k=1}^{K} x_{ikt} \le 1 \qquad \forall i,t \qquad (c5)$$

Binary variables showing the assignment of each task to the workstations in each period should be nonnegative.

$x_{ikt} \ge 0$ and integer $\qquad \forall i,k,t \qquad$ (c6)

Note that $x_{ikt} \le 1$ is satisfied by (c5)

The objective function tries to maximize the sum of net revenue obtained by each task done and is expressed as:

$$\text{Max} \sum_{i=1}^{m} \sum_{k=1}^{K} \sum_{t=1}^{s} P_i X_{ikt}$$

The Complete Disassembly Line Balancing model is:

$$\text{Max} \sum_{i=1}^{m} \sum_{k=1}^{K} \sum_{t=1}^{s} P_i X_{ikt}$$

Subject to (c1),(c2),(c3),(c4), (c5), (c6).

3.1.2 Model II

If there was no specified demand for any part in an optimal solution the same parts are produced in all cycles, i.e. periods hence the problem reduces to a single period problem so as to maximize the total profit.

We implement this idea, to produce our second model. We use two sub-models (Model A and Model B) to find an optimal solution.

<u>Model A</u>: Finds the minimum number of periods to satisfy all $\sum_i d_i$ units.

<u>Model B</u>: Finds the maximum profit for a single period, zero demand problem.

Formally Model A can be stated as follows:

<u>Decision Variables:</u>

$$Y_t = \begin{cases} 1, \text{if there is a production in period t} \\ 0, \text{otherwise} \end{cases}$$

X_{ikt} 's are defined in the original model.

<u>Objective Function:</u>

$$\text{Min } \sum_{t=1}^{s} Y_t - \varepsilon_P \sum_{i=1}^{m} \sum_{k=1}^{K} \sum_{t=1}^{s} P_i X_{ikt}$$

<u>Constraints:</u>

$$\sum_{i=1}^{m} \sum_{k=1}^{K} X_{ikt} \leq \mu * Y_t$$

μ is a big number and equals to the multiplication of task numbers and workstation numbers i. e. μ = M*K. Note μ is an upper bound on the optimal $\sum_{i=1}^{m} \sum_{k=1}^{K} X_{ikt}$ value. All other constraints are same with Model I.

The complete Model A is:

$$\text{Min} \sum_{t=1}^{s} Y_t - \varepsilon_P \sum_{i=1}^{m} \sum_{k=1}^{K} \sum_{t=1}^{s} P_i X_{ikt}$$

Subject to

$$\sum_{k=1}^{K} \sum_{t=1}^{s} x_{ikt} \geq d_i \qquad \forall i \qquad \text{(c1)}$$

$$\sum_{i=1}^{m} t_i x_{ikt} \leq CT \qquad \forall k,t \qquad \text{(c2)}$$

$$\sum_{k=1}^{k} x_{ikt} \leq \sum_{k=1}^{k} x_{lkt} \qquad \forall i,k,t,\ l \in PAND(i) \qquad \text{(c3)}$$

$$x_{ikt} \leq \sum_{h=1}^{K} \sum_{b \in POR(i)} x_{bht} \qquad \forall i,k,t \qquad \text{(c4)}$$

$$\sum_{k=1}^{K} x_{ikt} \leq 1 \qquad \forall i,t \qquad \text{(c5)}$$

$$\sum_{i=1}^{m} \sum_{k=1}^{K} X_{ikt} \leq \mu * Y_t \qquad \forall t \qquad \text{(c6)}$$

$$x_{ikt} \geq 0 \text{ and integer} \qquad \forall i,k,t \qquad \text{(c7)}$$

The objective function primarily minimizes the number of periods. Among the minimum number of periods solutions, it selects the one with the maximum profit. Production excess of demand is desirable, if it does not increase the number of periods.

The additional constraint activates Y_t, hence sets it to 1, if there is at least once task assigned to any workstation in period t. Note that $\sum_{i=1}^{m}\sum_{k=1}^{K} X_{ikt} \geq 1$ is upper bounded by μ . If $\sum_{i=1}^{m}\sum_{k=1}^{K} X_{ikt} \geq 1$ then Y_t=1 otherwise Y_t will be zero as we are minimizing $\sum_t Y_t$. The total number of periods is $\sum_t Y_t$, with at least one assignment.

The magnitude of ε_P is important in the sense that it should not increase the minimum $\sum_t Y_t$ en route to increasing total profit. ε_P should be set small enough such that;

$$\sum_{t=1} Y_t - \varepsilon_P * PT_{min} \le \sum_{t=1} Y_t + 1 - \varepsilon_P * PT_{max} \qquad (1)$$

Where PT_{min} = minimum possible total profit value i.e., a lower bound on the total profit. PT_{max} = maximum possible total profit value.

Hence, any solution with worse $\sum_t Y_t$ value, like $\sum_t Y_t + 1$, should not be favored even if it leads to the maximum improvement in the total profit value.

Expression (1) follows:

$$\varepsilon_P * PT_{max} - \varepsilon_P * PT_{min} \le 1 \qquad (2)$$

Hence, $\varepsilon_P \le \dfrac{1}{PT_{max} - PT_{min}}$

$PT_{max} \le T * \sum_i P_i d_i$ (When all products are produced in all periods with zero cost)

$PT_{min} \ge 0$ (When nothing is produced)

Hence, $\varepsilon_P = \dfrac{1}{T * \sum_i P_i d_i}$ to guarantee that among the minimum number of periods solutions we select the maximum profit solution. -

We let the objective function of Model A be Z_1, such that

$$Z_1 = Z_2 - \varepsilon_P \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{t=1}^{s} P_i X_{ikt} \text{ where } Z_2 = \sum_t Y_t \;.$$

Model A guarantees to satisfy all demand in Z_2 periods. In remaining (T- Z_2) periods, the repetitive cycles will be observed. To find an optimal solution in any cycle we solve Model B.

<u>Model B</u>

We drop subscript t from our decision variable X_{ikt}, as we are considering a single period. Our decision variable is X_{ik} where

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

Our objective function is $Z_3 = Max \sum_{i=1}^{m}\sum_{k=1}^{K} P_i X_{ik}$. We drop the demand constraint as we assume zero demand.

To find an optimal solution, we solve Models A and B and use the following solution:
First Z_2 periods X_{ikt} 's are found by Model A. In the last (T- Z_2) periods, $X_{ikt} = X_{ik}$ are found by Model B. Hence, the optimal objective function value Z becomes;

$$Z = (Z_2 - Z_1) / \varepsilon_P + (T - Z_2) * Z_3$$

where $(Z_2 - Z_1) / \varepsilon_P$ is the optimal solution of the first Z_2 periods and Z_3 is the optimal solution for a single period problem with zero demand.

In case an upper bound on the number of periods to satisfy all demand is known, i.e., UB (T_1) one may solve the original problem using UB (T_1), in place of T.

Then a single period model can be used to find the schedule for the rest of the cycles. Formally, let

Z_A = maximum profit for the original problem with UB (T_1) periods
Z_B = maximum profit for a single period problem with zero demand.

Then $Z = Z_A + (T - UB(T_1)) * Z_B$ is the optimal objective function value.

If UB (T_1) is small then Z_A can be found much easier than solving the original problem. Z_B is found very easy as it corresponds to a single period problem.

3.3. Complexity

Our DLBP is analogous to the multiple knapsack problems when there is a single period, the demands of the tasks are zero and there are no precedence relations. The analogy can be stated as follows:

The multiple knapsack problem (our problem) selects the items (tasks) of known weights (task times) and values (revenues) and assigns them to the knapsacks (workstations) of known capacities (cycle times) so as to maximize total value (revenue).

The multiple knapsack problem is NP-hard in the strong sense (Martello and Toth, 1990), so is our problem with additional complexity brought by multiple periods, part demands and precedence relations.

3.4. An Example Problem

Gupta (2002) presents a disassembly of a simple personal computer (PC) whose disassembly consists of eight tasks and eight parts. We use this example to illustrate the solutions of our models.

Tasks, related disassembled parts and parameters are shown in the table below:

Table 3.1: Parts and related parameters of the Example Problem

Task #	Definition	Times	Demands	Net Revenue
1	Removal of the top cover of the PC	12	1	21
2	Removal of the floppy drive	14	0	-15
3	Removal of the hard drive	12	4	14
4	Removal of the back plane	7	3	20
5	Removal of PCI cards	10	1	13
6	Removal of two RAM modules	2	2	3
7	Removal of the power unit	15	1	10
8	Removal of the motherboard	12	0	-14

The parameters are generated by our data generation scheme discussed in Chapter 5. We assume there are 4 workstations and 6 periods, i.e., disassembly products. We find cycle as CT= $1.5*\left\lceil \frac{\sum t_i}{K} \right\rceil$.

The precedence network is given below:

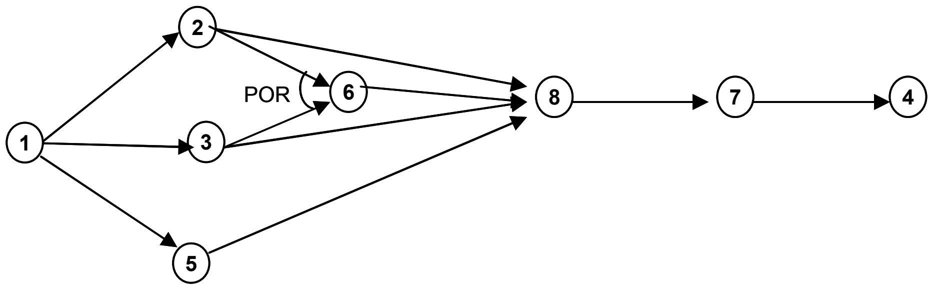


Figure 3.5: Precedence diagram of the Gupta (2002) PC Example

Table 3.2 tabulates the X_{ikt} values of a feasible solution and assignments of tasks to the workstations in each period. The table only includes the X_{ikt} values with value *"1"*. The other X_{ikt} values are *"0"*.

Table 3.2: Feasible solution for the Example Problem

Periods/Stations	1	2	3	4
1	1,2	5	3,6,8	4,7
2				
3		1	2	2
4	1,2,6		2,5,8	4,7
5	1,2,6	5	2,7,8	4
6				

The total net revenue of the above feasible solution is *"176"*.

Table 3.3 reports the X_{ikt} values *(the ones at value "1")* of an optimal solution and their assignmnets.

Table 3.3: Optimal solution for the Example Problem

Periods/Stations	1	2	3	4
1	1,3	2,5,6	8	4,7
2	1,3	2,5,6	7,8	4
3	1,3	2,5,6	7,8	4
4	1,3	2,5,6	7,8	4
5	1,3	2,5,6	7,8	4
6	1,3	2,5,6	7,8	4

The total net revenue of the optimal solution is *"312"*. The optimal solution is found by Model I. Note that the optimal solution has identical assignments for many periods. This is driving force for developing Model II. We now illustrate the solution with Model II.

The ε_P value to be used by Model A is found as follows:

$$\varepsilon_P = \frac{1}{T * \sum_i P_i d_i} = 0.003$$

$Z_1 = Z_2 - \varepsilon_P \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{t=1}^{s} P_i X_{ikt}$ where $Z_2 = \sum_t Y_t$.

The optimal $\sum_t Y_t$ value is found as "4" where the tasks are assigned to the workstations in periods 1, 3, 4 and 6. Periods 2 and 5 is not utilized due to minimization. Hence, Z_2=4 , Z_1=3.376 and $\varepsilon_P \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{t=1}^{s} P_i X_{ikt}$ =208. The optimal solution for Model A is given in Table 3.4.

Table 3.4: Optimal solution for Model A for the Example Problem

Periods/Stations	**1**	**2**	**3**	**4**
1	1,2,6	3,5	8	4,7
2				
3	1,3,6	2,5	8	4,7
4	1,5	2,3	6,7,8	4
5				
6	1,2,6	3	5,8	4,7

Model B solves a single period with maximization profit and sets all demands to *"0"*.

$Z_3 = Max \sum_{i=1}^{m}\sum_{k=1}^{K} P_i X_{ik}$ is found as *"52"*.

Note that the optimal objective function value Z can be written as;

$Z = (Z_2 - Z_1) / \varepsilon_P + (T - Z_2) * Z_3$

Thus,

Z = (4-3.376) / 0.003 + (6-4)*52 = 312

Note that 312 is the objective function value obtained from from Model I with T=6.

CHAPTER 4

THE DLBP PROBLEM

In this chapter we present our upper bounding procedures and the heuristic procedure that provides a lower bound.

4.1. Upper Bounds

We develop three upper bounds on the optimal value of the DLBP. These are namely *Upper Bound 1 (UB_1)*, *Upper Bound 2 (UB_2)* and *Upper Bound 3 (UB_3)*. UB_1 is obtained by relaxing the integrality constraints of Model I. UB_2 improves UB_1 through valid cuts. UB_3 takes its motivation from Model II.

4.1.1. Upper Bound 1 (UB1)

An optimal solution to any relaxation provides an upper bound on the optimal objective function value of our maximization problem.

Model I is solved to optimality by relaxing the integrality constraints on the binary X_{ikt} variables. The resulting optimal objective function value provides an upper bound on the optimal total net revenue.

The Linear Programming Relaxation (LPR) of the original problem is stated below:

$$Z_{LP} = \text{Max} \sum_{i=1}^{m} \sum_{k=1}^{K} \sum_{t=1}^{s} P_i X_{ikt}$$

Subject to

$$\sum_{k=1}^{K} \sum_{t=1}^{s} x_{ikt} \geq d_i \qquad \forall i \qquad \text{(c1)}$$

$$\sum_{i=1}^{m} t_i x_{ikt} \leq CT \qquad \forall k,t \qquad \text{(c2)}$$

$$\sum_{k=1}^{K} x_{ikt} \leq \sum_{k=1}^{K} x_{lkt} \qquad \forall i,k,t,\ l \in PAND(i) \qquad \text{(c3)}$$

$$x_{ikt} \leq \sum_{h=1}^{K} \sum_{b \in POR(i)} x_{bht} \qquad \forall i,k,t \qquad \text{(c4)}$$

$$\sum_{k=1}^{K} x_{ikt} \leq 1 \qquad \forall i,t \qquad \text{(c5)}$$

$$x_{ikt} \geq 0 \qquad \forall i,k,t \qquad \text{(c6)}$$

Note that the only difference between the original model and its LP Relaxation is the integrality requirement on X_{ikt} values. This follows, Z_{LP} is an upper bound on the optimal objective function value Z^*.

Optimal solution to the LPR of the problem discussed in Chapter 3 (Gupta's PC example) is as follows:

$X_{441}= X_{442}= X_{443}= X_{444}= X_{445}= X_{446}= X_{621}= X_{624}= X_{625}= X_{626}= X_{741}= X_{742}= X_{743}= X_{744}= X_{745}= X_{746}= 1$

$X_{114}= X_{214}= X_{514}= X_{633}= X_{642}=0.03$, $X_{332}= X_{832} =0.14$, $X_{341}= X_{841}= X_{541}=0.15$, $X_{331}= X_{531}=0.35$, $X_{342}= X_{842}=0.37$, $X_{124}= X_{224}= X_{524}=0.47$,

$X_{122}= X_{123}= X_{222}= X_{223}= X_{322}= X_{323}= X_{522}= X_{523}= X_{822}= X_{823}=0.48$,

$X_{115}=X_{116}=X_{121}=X_{131}=X_{134}=X_{135}=X_{136}=X_{215}=X_{216}=X_{221}=X_{231}=X_{234}=X_{235}=X_{236}=$

X_{315}=X_{316}=X_{321}=X_{324}= X_{334}= X_{335}= X_{336}=X_{516}=X_{521}=X_{525}=X_{534}=X_{535}=X_{536}=X_{824}= X_{825}=X_{826} = X_{834}= X_{835}=X_{836}= 0.50, X_{132}= X_{133}= X_{232}= X_{233}= X_{333}= X_{532}= X_{533}= X_{833}= 0.52, X_{831}= 0.85, X_{622}= X_{623}= 0.97

All other X_{ikt} s are *"0"* and $Z_{LP} = 312$. Note that, out of 87 variables, 16 of them is found as "1" and 71 of them is found fractional by the optimal LP Relaxation.

4.1.2. Upper Bound 2 (UB2)

UB_2 is found by adding valid cuts to the LP relaxed problem, i.e., UB_1.

The valid cuts are found by investigating the properties of all or at least one of the optimal solutions that may not be satisfied by the optimal LP Relaxation.

- **Fixing the Demands (CUT1)**

Theorem 1: There exists an optimal solution in which *dj* units of *part j* is released in the first *dj* periods.
Proof: Assume an optimal solution in which the first *dj* units are produced in period $t_1, t_2, ..., t_{dj}$. (t_i is the period where ith unit of *part j* is released). If the task contents of any two periods are changed, i.e., X_{ikta} is set to X_{iktb} and X_{iktb} is set to X_{ikta} for all i and t_k for any two periods t_a and t_b, then the optimal objective function value is not affected. Assume an optimal solution in which *part j* is released in periods $t_{a+1}, t_{a+2}, ...t_{a+dj}$. Setting $X_{jkr} = X_{jk(ta+r)}$ for all j, $r <= d_j$ and $X_{jk(ta+r)} = X_{jkr}$ does not change the objective function value. This follows, there is an optimal solution in which *dj* units of *part j* is released in the first *dj* periods. □

By using the result of Theorem 1, one can fix a single *task j* to first d_j periods. En route to maximizing the number of periods fixed to a particular task, we select the task having the maximum demand. Hence we fix part releasing *task r* such that $d_r = Max\ \{d_j\}$ to the first d_r periods.

The equation set that supports Theorem1 is stated as below:

$$\sum_{k=1}^{K} \sum_{t=1}^{dr} x_{rkt} = d_r$$

As x_{rkt} =0 or 1, there can be at most one release for *task r* in each period. By the above relation, we guarantee that there is exactly one release for *task r* in each period *t*, where $1 \leq t \leq dr$.

We add Cut1 to the relaxed problem and compare objective function value and number of fractional values with and without Cut1 (Pure LPR)

Table 4.1: The comparison of Pure LPR and LPR with CUT1

	# of fractions	**CPU Time**
Pure_LPR	73	0.06
LPR(Cut1)	60	0.05

As can be seen from the LPR solution, *Task 3* is split into "14" among "4" workstations. X_{3kt} variables have the following fractional values:
X_{332}=0.14, X_{341}=0.15, X_{331}=0.35, X_{342}=0.37, X_{322}= X_{323}=0.48, X_{315}=X_{316}=X_{321} X_{324}= X_{334}= X_{335}= X_{336}= 0.50, X_{333}= 0.52.
By adding *Cut1*, the following X_{3kt} values are obtained:
X_{311}=0.91, X_{312}=0.86, X_{313}=0.64, X_{314}=0.75, X_{321}=0.09, X_{322}=0.14, X_{333} = 0.36, X_{334} = 0.25, X_{325} = X_{336} = 1.
X_{3kt} variables have a total of *8 fractional values*, and two variables at *"1"*.

Fixing demand decreases the fractional values and decreases the CPU time, as it eliminates many alternate optimal solutions.

- **Lower Bound on workstation positions (CUT2)**

Cut2 provides a lower bound on the workstation number that can reside any defined task.

Theorem 2: In all optimal solutions, *task j* cannot be assigned to workstations *1,2,..,E_J-1* where $E_J = \left\lceil \dfrac{(\sum_{i \in Pj} t_i + t_J)}{CT} \right\rceil$ and P_j is the set of "*AND*" predecessors.

Proof: *Task j* cannot start before its predecessors hence should wait at least $\sum_{i \in Pj} t_i$ units to start and $\sum_{i \in Pj} t_i + t_J$ units to complete. When task splitting is allowed and all other tasks are ignored, $\sum_{i \in Pj} t_i + t_J$ units require $\left\lceil (\sum_{i \in Pj} t_i + t_J) / CT \right\rceil$ workstations. When task splitting is not allowed and other tasks are considered $\left\lceil (\sum_{i \in Pj} t_i + t_J) / CT \right\rceil$ becomes a lower bound on the earliest workstation for *task j.* □

The following constraint set is used to support Theorem 2.

$$\sum_{k=1}^{Ej-1} \sum_{t=1}^{s} X_{jkt} = 0 \qquad \forall j$$

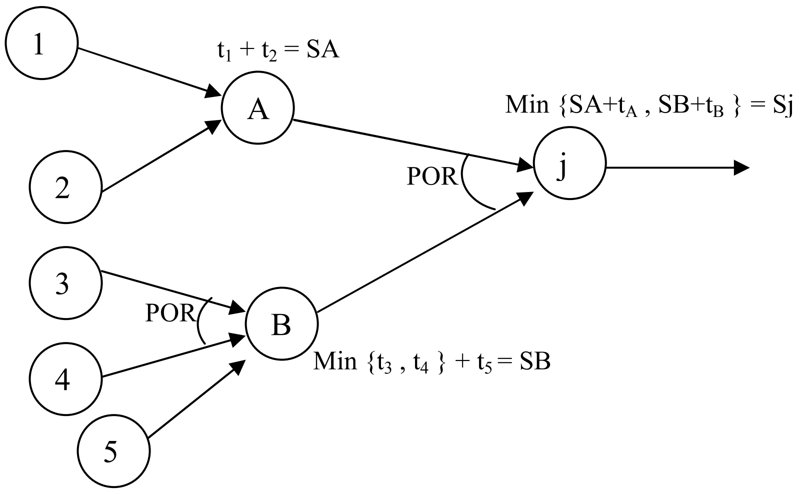
By the above equation, we guarantee that task j is not assigned to workstations *1,2,..,*E_J*-1* in any period.

The theorem gives a lower bound on the station number that resides task j. It ignores "POR" relations. We modify this lower bound to include the "POR" relations while preserving the validity of the bound. In doing so, starting from the source node we replace the "POR" relations with "AND" relations and take the minimum task time of a "POR" relation as the task time of the "AND" relation. As minimum times are used in place of exact times, the sum of the processing time $\sum_{i \in PORj} t_i \geq \sum_{i \in \overline{PORj}} t_i$ where $\overline{PORj}$ is the PORj relation replaced by AND relation using minimum task time.

The modification of Theorem 2 with $\overline{PORj}$ inclusion replaces E_j by;

$$E_J = \left\lceil \frac{(\sum_{i \in (Pj \cup \overline{PORj})} t_i + t_J)}{CT} \right\rceil$$

The following example illustrates $\sum_{i \in \overline{PORj}} t_i$ computations:



1
2
3
4
5
A
B
j
$t_1 + t_2 = SA$
Min {SA+t_A , SB+t_B } = Sj
POR
POR
Min {t_3 , t_4 } + t_5 = SB

Figure 4.1: An example illustrating Theorem 2

POR_b defined by *{3, 4}* is replaced by $\overline{POR_b}$ with task time *Min {t_3, t_4}*.

(t_1 + t_2) is the minimum total processing load to start *task A (SA)* and *Min {t_3,t_4}*+ t_5 is the minimum total processing load to start *task B (SB).* This follows;

Min {(t_1 + t_2+ t_A), (min {t_3, t_4} + t_5 + t_B) } is the minimum total processing load to start *task j*. Note that $E_J = \left\lceil \frac{\text{Minimum total processing load to start task } j + t_j}{CT} \right\rceil$.

Hence, in this example

$$E_J = \left\lceil \frac{\text{Min}\{(t_1 + t_2 + t_A), (\min\{t_3, t_4\} + t_5 + t_B)\} + t_j}{CT} \right\rceil.$$

We use the below network to illustrate *Theorem 2*. The tasks are given on the network. We use *K=4, CT=63, T=80*. E_j value is calculated as follows for task 13;

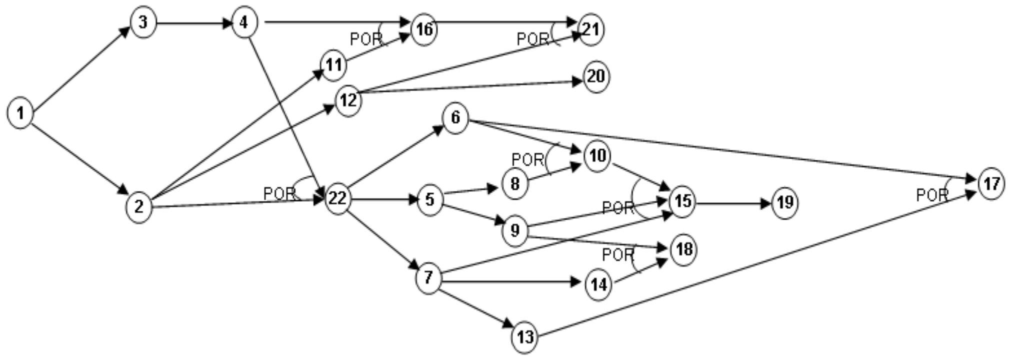


Figure 4.2: Precedence diagram of the Lambert's ball-point pen example (22 tasks)

$$E_{13} = \left\lceil \frac{\text{Min}\,\{(t_1 + t_3 + t_4), (t_1 + t_2)\} + t_{22} + t_7 + t_{13}}{CT} \right\rceil$$

$$E_{13} = \left\lceil \frac{\text{Min}\,\{(13 + 1 + 6), (13 + 7)\} + 19 + 15 + 17}{63} \right\rceil$$

$$E_{13} = \left\lceil \frac{71}{63} \right\rceil = 2$$

Hence, *task 13* can not be assigned to the first workstation in any period.

The optimal LPR assigns *task 13* to workstation *"1"* at *62* periods. There are *53, 50 and 39* fractional variables for the assignment of task 13 to *workstations "2", "3", and "4"* respectively. Hence, there are a total of *142* fractional variables.

After introducing Cut2 to the LPR, there is *no assignment to workstation "1"*. The number of fractional variables for *workstations "2", "3", and "4"* become *28, 26 and 40* respectively.

Note that the number of fractional variables is reduced to *"94"* from *"142"* for task 13.

The objective function value and number of fractional variables of LPR with Cut2 and Pure_LPR are compared, and tabulated below.

Table 4.2: The comparison of Pure LPR and LPR with CUT2

	Objective function value	**# of fractions**	**CPU Time**
Pure_LPR	26.796	3.643	1.25
LPR(Cut2)	26.076	1.702	1.04

Note from the table that by Cut2, the *number of fractional variables* is almost hauled and the *CPU time* is reduced to *1.04 seconds* from *1.25 seconds.*

- **Projecting Demands (CUT3)**

The demand figures are related only to the part releasing tasks. The optimal solution guarantees that the total number of part releasing tasks assigned is never greater than the number of its predecessor tasks assigned. However the optimal LP relaxation may not satisfy this property due to the allowed partial assignments. Recognizing this fact, we project the demands of the part releasing tasks to all other tasks, and impose the following constraints.

i. "AND" precedence reations

Consider the following network in Figure 4.3:

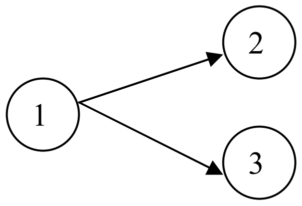


Figure 4.3: Projecting Demands for "AND" precedence relations

The total number of *task 1* assigned should not be smaller than the total amount of *task 2* and *task 3* assigned. Hence the following relation should hold:

$$\sum_{k}^{K}\sum_{t}^{s} x_{1kt} \geq \max\left\{\sum_{k}^{K}\sum_{t}^{s} x_{2kt}, \sum_{k}^{K}\sum_{t}^{s} x_{3kt}\right\}$$

In general; $\sum_{k}^{K}\sum_{t}^{s} x_{ikt} \geq \max_{j \in S(i)} \left\{\sum_{k}^{K}\sum_{t}^{s} x_{jkt}\right\}$ where *S(i)* is the set of immediate successors of *task i.*

The relation can be linearized as; $\sum_{k}^{K}\sum_{t}^{s} x_{ikt} \geq \sum_{k}^{K}\sum_{t}^{s} x_{jkt} \qquad \forall j \in S(i)$

ii. "POR" precedence reations

Consider the following network:

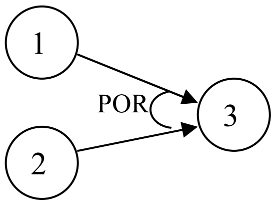


Figure 4.4: Projecting Demands for "POR" precedence relations

Note that *Task 3* requires the assignment of *task 1* or *task 2*. This follows the total amount of *task 1* and *task 2* assigned should not be less than the number of *task 3* assigned. This relation is imposed by the following constraint:

$$\sum_{k}^{K}\sum_{t}^{s} x_{1kt} + \sum_{k}^{K}\sum_{t}^{s} x_{2kt} \geq \sum_{k}^{K}\sum_{t}^{s} x_{3kt}$$

In general; $\sum_{j \in POR(i)} \sum_{k}^{K}\sum_{t}^{s} x_{jkt} \geq \sum_{k}^{K}\sum_{t}^{s} x_{ikt}$

The following network explains more complex relations:

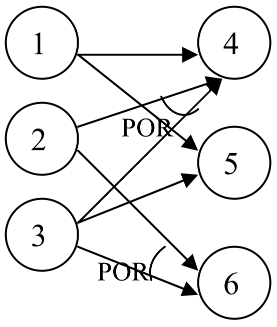


Figure 4.5: Projecting Demands for complex relations

Constraint sets used for this complex network are as follows:

$$\sum_{k}^{K}\sum_{t}^{s} x_{1kt} \geq \max\left\{\sum_{k}^{K}\sum_{t}^{s} x_{4kt}, \sum_{k}^{K}\sum_{t}^{s} x_{5kt}\right\}$$

$$\sum_{k}^{K}\sum_{t}^{s} x_{3kt} \geq \sum_{k}^{K}\sum_{t}^{s} x_{5kt}$$

$$\sum_{k}^{K}\sum_{t}^{s} x_{2kt} + \sum_{k}^{K}\sum_{t}^{s} x_{3kt} \geq \sum_{k}^{K}\sum_{t}^{s} x_{4kt}$$

$$\sum_{k}^{K}\sum_{t}^{s} x_{2kt} + \sum_{k}^{K}\sum_{t}^{s} x_{3kt} \geq \sum_{k}^{K}\sum_{t}^{s} x_{6kt}$$

We use the network given in Figure 4.2 with *K=4, CT=63, T=80* to illustrate the demand projecting cuts. As it can be seen from the figure, Task 22 is the immediate predecessor of Tasks 5, 6 and 7. Hence, the constraints added to the model for Task 22 are:

$$\sum_{k}^{K}\sum_{t}^{s} x_{22kt} \geq \sum_{k}^{K}\sum_{t}^{s} x_{5kt}$$

$$\sum_{k}^{K}\sum_{t}^{s} x_{22kt} \geq \sum_{k}^{K}\sum_{t}^{s} x_{6kt}$$

$$\sum_{k}^{K}\sum_{t}^{s} x_{22kt} \geq \sum_{k}^{K}\sum_{t}^{s} x_{7kt}$$

The optimal LP relaxation does not satisfy this property due to the allowed partial assignments. The solutions are

$$\sum_{k}^{K}\sum_{t}^{s} x_{22kt} = 20\ ,\ \sum_{k}^{K}\sum_{t}^{s} x_{5kt} = 80,\ \sum_{k}^{K}\sum_{t}^{s} x_{6kt} = 80,\ \sum_{k}^{K}\sum_{t}^{s} x_{7kt} = 80$$

and the added constraint sets are violated.

The optimal LPR solution with Cut3 gives the following result;

$$\sum_{k}^{K}\sum_{t}^{s} x_{22kt} = 80\ ,\ \sum_{k}^{K}\sum_{t}^{s} x_{5kt} = 80,\ \sum_{k}^{K}\sum_{t}^{s} x_{6kt} = 80,\ \sum_{k}^{K}\sum_{t}^{s} x_{7kt} = 80$$

satisfying the related constraints added and the precedence relationships.

By using cut3, one can guarantee that the total amount of tasks assigned is never greater than the amount of its predecessor tasks assigned. The objective function value and number of fractional variables of LPR with cut3 and Pure_LPR are compared and tabulated below:

Table 4.3: The comparison of Pure LPR and LPR with CUT3

	Objective function value	**# of fractions**	**CPU Time**
Pure_LPR	26.796	3.643	1.25
LPR(Cut3)	26.076	1.979	1.20

Note from the table that, the *number of fractional variables* reduces to *"1.979"* from *"8.643"*. Moreover the *CPU time* reduces to *1.20* from *1.25* and there is a slight improvement in the objective function value.

- **Existence of Feasible Solution-Ranking Heuristic (CUT4)**

Our last cut uses the results of the following two theorems.

Theorem 3: If there exists a feasible schedule that processes all tasks that are *not successors of task j and task j* in r workstations then there exists an optimal schedule in which task j is processed in workstations *"1" through "r".*

Proof: Assume *task j* is assigned to *workstation "k"* such that $k \leq r$, and workstations r+1 through K process the *successors of task j*. Taking *task j* from workstation k and placing to workstation *"l"* such that $l \geq$ r+1 cannot increase the total revenue as such a replacement cannot allow more task assignments. This is due to the fact that all tasks in workstations r+1 through K are successors of *task j,* hence cannot be processed in workstations 1 through r, once task j is processed later. This follows, there exists an optimal solution in which *task j* is processed in the *first r workstations*. ☐

Theorem 4: If there exists a feasible schedule that processes all tasks that are *not predecessors of task j and task j* in the last r workstations, then there exists an optimal schedule in which *task j* is processed in workstations "K-r+1" through "K".

Proof: Assume *task j* is assigned to *workstation "k"* such that k$\geq$ K-r+1 and workstations 1 through K-r process the predecessors of task j. Taking task j from workstation k and placing it to workstation "*l*" such that $l \leq$K- r cannot increase the total revenue as such a replacement cannot allow more task assignments. This is due to the fact that all tasks in workstations 1 through K-r are predecessors of *task j,* hence cannot be processed in workstations K-r+1 through K, once task j is processed earlier. This follows, there exists an optimal solution in which *task j* is processed in the *last r workstations*. ☐

To find feasible solutions to implement the theorems, we use the following *heuristic procedure*:

Let *S* be the set of tasks that should be assigned to *K* workstations. Order the tasks in *set S* according to their nondecreasing order of task times. Take the tasks from the order starting from the first *feasible task*. A task is feasible if its inclusion to the current workstation *does not violate the precedence relations and cycle time constraints*. If no such task exists close the current workstation and open a new one. Stop when all jobs in set S is assigned. Let *num(S)* be the resulting number of workstations.

To implement Theorem 3 we set $S{=}S_1$ where S_1 is the set of all tasks except the successors of *task j*. To implement Theorem 4 we set $S{=}S_2$ where S_2 is the set of all tasks except the predecessors of *task j.* After $num(S_1)$ and $num(S_2)$ are obtained, implementing the heuristic for sets S_1 and S_2, we include one of the following two cuts:

i. $\sum_{t} \sum_{k=num(s1)+1}^{K} X_{jkt} = 0 \qquad \forall j$

This cut supports the result of *Theorem 3.*

ii. $\sum_{t} \sum_{k=1}^{K-num(s2)} X_{jkt} = 0 \qquad \forall j$

This cut supports the result of *Theorem 4.*

i. and ii. cannot be included to the LPR simultaneously as they may lead to infeasible solution. To obtain a feasible solution, we select one of them using the following rule:

Rule: Use *"i"* if $K\text{-}num(S_1) \geq K\text{-}num(S_2)$, else use *"ii"*.

The idea behind the rule is to prevent assignments to more workstations, hence use stronger cut.

We illustrate the Cut4 through our previous network with same data. The tasks in the problem are ranked as in the table below:

Table 4.4: Related data for CUT 4

Rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Tasks	1	3	4	2	12	11	20	16	21	22	6	5	8	10	7	14	13	17	9	15	19	18
Task times	13	1	6	7	12	14	18	18	16	19	5	7	8	3	15	4	17	15	17	11	4	19
K-num(S_1)	3	3	3	3	1	1	1	1	0	1	3	3	1	1	0	2	0	0	0	2	2	2
K-num(S_2)	0	0	0	0	2	2	2	2	3	2	0	0	2	2	3	1	2	3	3	0	1	1
Num selected	3	3	3	3	2	2	2	2	3	2	3	3	2	2	3	2	2	3	3	2	2	2

The tasks assigned to the workstations via first and second rules are shown in the table 4.5.

Table 4.5: Assignment of tasks according to procedures

Stations	**1**	**2**	**3**	**4**
1st rule_Tasks	1,2,3,4,11,12	16,20,21,22	5,6,7,8,10,13,14	9,15,17,18,19
2nd rule_Tasks	1,2,3,4,11,12,20	16,21,22	5,6,7,8,10,13,14,17	9,15,18,19

Note that the information given in the table entails that;

- Tasks cannot be assigned after workstation k *in any period t* where k is the workstation that task i is assigned by the *1st rule.*
- Tasks cannot be assigned before the workstation k *in any period t* where k is the workstation that task i is assigned by the *2nd rule.*

For example, *task 19* cannot be assigned to workstations *"1", "2" and "3"* with both rules (even the selected rule is the 2nd one). The resulting solution for *task 19* satisfies the constraint;

$$\sum_{t} \sum_{k=1}^{3} X_{19\_k\_t} = 0$$

and assigns task 19 to only workstation "4" such as; $X_{19\_4\_49}=X_{19\_4\_65}=1$.

The solution of Pure LP gives the assignment of task 19 as; $X_{19\_1\_1}=0.75$, $X_{19\_1\_34}=0.25$, $X_{19\_4\_73}=1$, hence the condition is not satisfied.

The optimal solution after adding *Cut4* is shown in the table below:

Table 4.6:The comparison of Pure LPR and LPR with CUT4

	Objective function value	**# of fractions**	**CPU Time**
Pure_LPR	26.796	3.643	1.25
LPR(Cut4)	26.076	326	0.50

Note that *the number of fractions* and *CPU time* decrease drastically by *Cut4.*

UB_2 is the total revenue returned by LP after introducing all four cuts.

4.1.3. Upper Bound 3 (UB3)

To obtain UB_3, we use the idea used to construct *Model II.* In *Model A* in place of minimizing number of periods, we obtain an estimate on the minimum number of periods to satisfy all demand. Our aim here is to get rid of the Y_t binary variables.

Firstly, a lower bound is found on the minimum number of periods *(LB (T))* and it is used "as is" or increased iteratively till a feasible solution is reached. *Model B* is used without any change.

We find *(LB (T))* through the following theorem:

Theorem 5: The demand cannot be satisfied in less than *LB (T)* periods where $\mathrm{LB(T)} = \left\lceil \frac{\sum_j p_j * dnew_J}{K * CT} \right\rceil$ and $dnew_J$ is the projected demand of *task j.*

Proof: The total processing requirement to produce all demand is $\sum_j p_j * dnew_J$ where $dnew_J$ is the projected demand of *task j.* In each period there are *K* workstations and each workstation is available for *CT* time units, hence a total processing availability by a single period is *K*CT* time units. $\left\lceil \frac{\sum_j p_j * dnew_J}{K * CT} \right\rceil$ is the number of periods to satisfy all demand if task splitting between the periods and workstations are allowed. As no task splitting is allowed $\left\lceil \frac{\sum_j p_j * dnew_J}{K * CT} \right\rceil$ is a lower bound on the number of periods *(LB (T))*. □

We use the network in Figure 4.2 with *K=4 and CT=63* to illustrate the implementation of Theorem 3. *LB (T)* is found as *"3"* by the help of the Theorem3. We find d_{new} values as shown in the table below:

Table 4.7: Projected demands of tasks ($\mathbf{d_{new}}$)

Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$\mathbf{d_{new\,(j)}}$	4	4	4	4	4	4	4	3	4	4	4	4	4	3	2	1	0	4	2	1	0	4
Task times	13	7	1	6	7	5	15	8	17	3	14	12	17	4	11	18	15	19	4	18	16	19

We obtain LB (T) = $\left\lceil \frac{13*4+7*4+\ldots+19*4}{4*63} \right\rceil$ =3, using *Theorem 3*.

Then model with cuts (cut1, cut2, cut3 and cut4) is solved by setting *T to "3"*. With *T=3* we could not obtain a feasible solution, then we set *T=4*. With *T=4* there is no feasible solution as well. Hence with *T=5* LP model with cuts is solved and a feasible solution is obtained. This means that the minimum number of periods to satisfy all demand is *"5"* periods *(UB (T) = 5)*.

Finally the maximizing revenue model with cuts is solved for a single period, i.e., *T=1*, and *all zero demand*. The resulting solutions are shown below:

Table 4.8: Optimal solutions with T=5 and T=1

	Objective function value	CPU Time
T=5	1.626	0.05
T=1	326	0.03

An upper bound on the number of periods to satisfy all demand is known, i.e., *UB (T) =5* and the problem is solved using *UB (T)=5,* in place of *T=80.* Then a single period model can be used to find the schedule for all remaining *T – UB (T) = 80-5 = 75 periods*. Let

Z_A = maximum revenue for the original LP problem with *UB (T) =5* periods
Z_B = maximum revenue for a single period problem with zero demand. *(T =1)*

Then $UB_3 = Z_A + (T - UB(T)) * Z_B$ is the optimal objective function value. This follows;
$UB_3 = 1.626 + (80 - 5) * 326 = 26.076$

Note that $UB_2 = UB_3$. Since UB (T) =5 is small, Z_A can be found much easier than solving the problem with *T=80.* Z_B is found very easy as it considers a single period problem. Hence one should expect to obtain UB_3 much quicker than UB_2. We obtain UB_2 *and* UB_3 values for our example in *0.45* and *0.08* seconds respectively.

4.2. Lower Bound

The satisfactory behavior of UB_3 in terms of its solution time and number of fractions it produces motivated us to produce a feasible solution around UB_3 solution. We use UB_3 as a starting solution en route to finding a feasible solution to our problem.

Our lower bound first solves UB_3, then fixes its variables that are assigned to “1” and obtains a reduced problem. The reduced problem has only the partially assigned or unassigned tasks of UB_3 solution. As the partial assignments are quite low compared to the original variables, the reduced problem is very small in size. As our Mixed Integer Linear Programming (MILP) can handle small-sized problems very quickly, we prefer to use it to find an optimal solution for not completely assigned tasks.

After fixing the variables to "1", the resulting solution by MILP may be infeasible due to the following two reasons:

i. The cycle time constraint cannot be satisfied
ii. The precedence relations cannot be satisfied

If the cycle time constraint cannot be satisfied, then by increasing the number of periods, a feasible assignment can be reached. However if the precedence relations cannot be satisfied, there is no way to reach to a feasible solution without changing the fixed tasks.

If (i) holds then we increase the number of periods one by one until we obtain a feasible solution. If (ii) holds the tasks whose fixings lead to violation of the precedence relations, are set to zero, and the MILP is resolved. Below we give the stepwise description of our lower bounding procedure:

Step1: Solve the LP Relaxation with cuts using minimum number of periods. Fix the variables that are assigned to "1" in the relaxed solution.

Step2: Solve the MILP for the unassigned tasks considering the assignments made in Step1. If the solution is feasible, STOP.

Step3: If the infeasibility is caused by the cycle time constraints, let $T=T+1$, go to Step1. If the infeasibility is caused by the precedence relations, unassign the related tasks, go to *step2.*

After finding the feasible solution with a feasible period number, namely *UB(T),* MILP is solved to maximize revenue with zero demand and a single period. Let,

Z_A = maximum revenue resulting by feasible solution of MILP with *UB(T).*
Z_B = maximum revenue with MILP for a single period problem with zero demand. *(T =1)*

Then LB= Z_A+ (T – UB (T))* Z_B is the optimal objective function value for *lower bound problem.*

To illustrate our lower bound, we use a numerical example with *8 tasks* and we assume *K=4, CT=31 and T=6*. The precedence network is given in Figure 3.5 and related parameters are tabulated in Table 3.1. For *UB_3; LB (T)* and *UB (T)* values are found as *"2"* and *"5"* respectively. Then the following steps of the lower bound are executed;

Step1: LP Relaxation with cuts and *T=UB (T) =5* is solved and the variables that receive *value "1"* in the solution are fixed.
Step2: The MILP for the unassigned tasks considering the assignments made in *Step1* is solved. Variables (tasks) that are assigned to "1" in the relaxed solution are shown in Table 4.9 and the loads of the workstations are shown in Table 4.10. The fractional variables are; X_{311}= X_{312}= X_{313}= X_{314}=0.64, X_{315}=0.75, X_{321}= X_{322}= X_{323}= X_{324}=0.36, X_{325}=0.25.

Table 4.9: Tasks that are assigned to "1" in the relaxed problem (T=5)

Periods/Stations	**1**	**2**	**3**	**4**
1	1,5,6	2,8		4,7
2	1,5,6	2,8		4,7
3	1,5,6	2,8		4,7
4	1,5,6	2,8		4,7
5	1,5	2,6,8		4,7

Table 4.10: The loads of the workstations after full assignments (T=5)

Periods/Stations	**1**	**2**	**3**	**4**
1	24	26		22
2	24	26		22
3	24	26		22
4	24	26		22
5	22	28		22

The resulting MILP solution for the unassigned tasks is infeasible due to the precedence relations. Note that the returned solution has an empty workstation and this implies that the infeasibility is caused by the precedence relations. Then we continued with step 3. *Task 8* is fully assigned to *workstation 2* in every period by the help of fractional assignments of its immediate *predecessors 3 and 6.* When these fractional assignments are ignored and only the variables that are assigned to "1" are considered for the MILP, the remaining cycle time for the predecessors of task 8 is inadequate. Lets consider first period; as *CT=31* and remaining cycle time for *workstation 1 = 31-24= 7,* remaining cycle time for *workstation 2 = 31-26= 5*, *task 3* having processing time *"12"* cannot be assigned to *workstations 1 and 2*. Therefore the infeasibility with these assignments is inevitable.

Step3: Since the infeasibility is caused by the precedence relations, we ignore the assignment of *task 8* to *workstation 2* in each period and repeat step 2.
Step2: The MILP is solved with the fractional tasks of Step1 and freed tasks of step 3. We obtain a feasible solution *with T=5*, and stop. The resulting assignments are shown in Table 4.11.

Table 4.11: MILP solution (T=5)

Periods/Stations	1	2	3	4
1	1,5	2,3,6	8	4,7
2	1,5	2,6	3,8	4,7
3	1,5	2,3,6	8	4,7
4	1,5	2,6	3,8	4,7
5	1,5	2,3,6	8	4,7

The objective function value Z_A, is found as *260* with *UB(T)=5*. Then we solve a single period MILP with zero demand for maximizing revenue. The objective function value Z_B, is obtained as *52*. To find *LB*, we use;

LB= Z_A+ (T – UB(T))* Z_B

LB= 260+ (6 – 5)* 52 =312

As a result, *lower bound* for the original problem is obtained as *"312"*.

Another numerical example is used to illustrate the infeasibility due to the cycle time constraints. The example has 22 tasks with the parameters shown in Table 4.12 and *K=2, CT=115 and T=80*. Firstly, *LB (T) and UB (T)* values are found as *"3" and "5"* respectively for *UB_3*. Then the following steps are followed;

Table 4.12: Data for the second example instance

Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Times	14	1	6	5	8	18	14	14	18	8	8	18	6	8	11	7	14	15	14	15	3	4
Demands	1	1	3	1	1	1	1	4	4	2	3	3	4	4	3	3	1	1	2	3	3	3
Net Rev.	28	23	-1	27	22	30	13	5	19	-3	37	13	42	33	25	32	27	15	10	4	32	10

Step1: The LP Relaxation with cuts and *T=5*. The variables that are assigned to "1" in the relaxed solution, are fixed.

Step2: The MILP for the unassigned tasks considering the assignments made in Step1 is solved. Since the solution is infeasible, we proceed to step 3.

Step3: The infeasibility is caused by the cycle time constraints. The solution of the relaxed problem with *UB (T) =5* gives all full assignments, except *Task 10.* The fractional values for *Task 10* are; $X_{10\_2\_1}$= $X_{10\_2\_5}$=0.88, $X_{10\_2\_2}$=0.24. The loads of workstations with the fully assigned tasks are shown in Table 4.13.

Table 4.13: Loads of the with full assignments (T=5)

Periods/Stations	**1**	**2**
1	113	108
2	113	108
3	113	108
4	113	108
5	113	108

Note that the *remaining cycle times* for *Workstations 1 and 2* in each period is *"115-113=2"* and *"115-108=7"* respectively. The processing time of *Task 10* is *8 units*; hence the workstations cannot process Task 10 with their remaining times. Therefore, to get a feasible solution, *T* has to be increased. We let *T=T+1*, go to Step1.

The period number with a feasible solution is found as *UB (T) =7* and Z_A*=3116.* The assignments of *Task 10* are; $X_{10\_1\_6}= X_{10\_2\_7}= 1$. Then we solve a single period MILP with zero demand for maximizing revenue. Objective function is obtained as Z_B *=446*. To find *LB*, we use;

LB= Z_A+ (T – UB (T))* Z_B

LB= 3116+ (80 – 7)* 446 =35.674.

Our *lower bound* for the original problem is *"35.674"*.

CHAPTER 5

COMPUTATIONAL EXPERIMENT

In this chapter we present the results of our computational experiment. We first discuss our data generation scheme then state our performance measures. Finally we evaluate the computational results.

5.1. Data Generation

In this section, we present the network and parameter generation schemes.

5.1.1. Network Generation

We start with small networks and then form bigger networks. To generate big networks we take small networks from the literature, combine them and put additional arcs.

1st Network (22tasks)

Lambert (1997) illustrates a 10-part ball-point pen example with its assembly as shown in Figure 5.1.

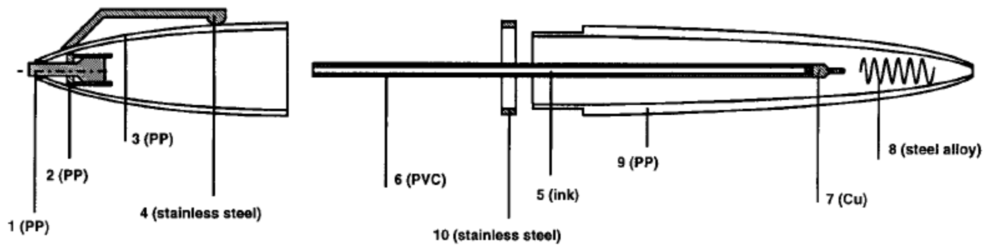


Figure 5.1: 10 part ball-point pen example (Lambert, 1997)

The disassembly operations on a disassembly graph with 20 tasks are demonstrated in Figure 5.2.

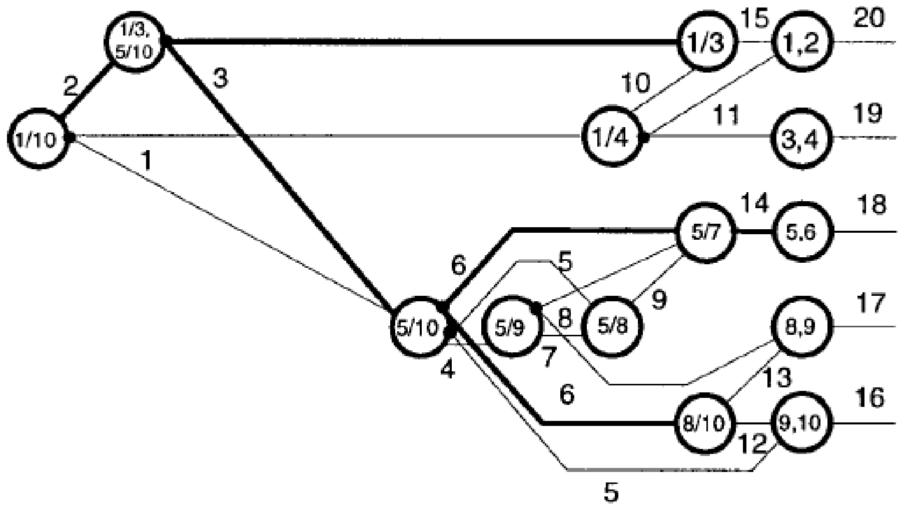


Figure 5.2: Disassembly graph of the 10 part ball-point pen example (Lambert, 1997)

We transform this disassembly graph into a precedence diagram by adding a dummy task (task 22). The resulting diagram is shown in the figure below:

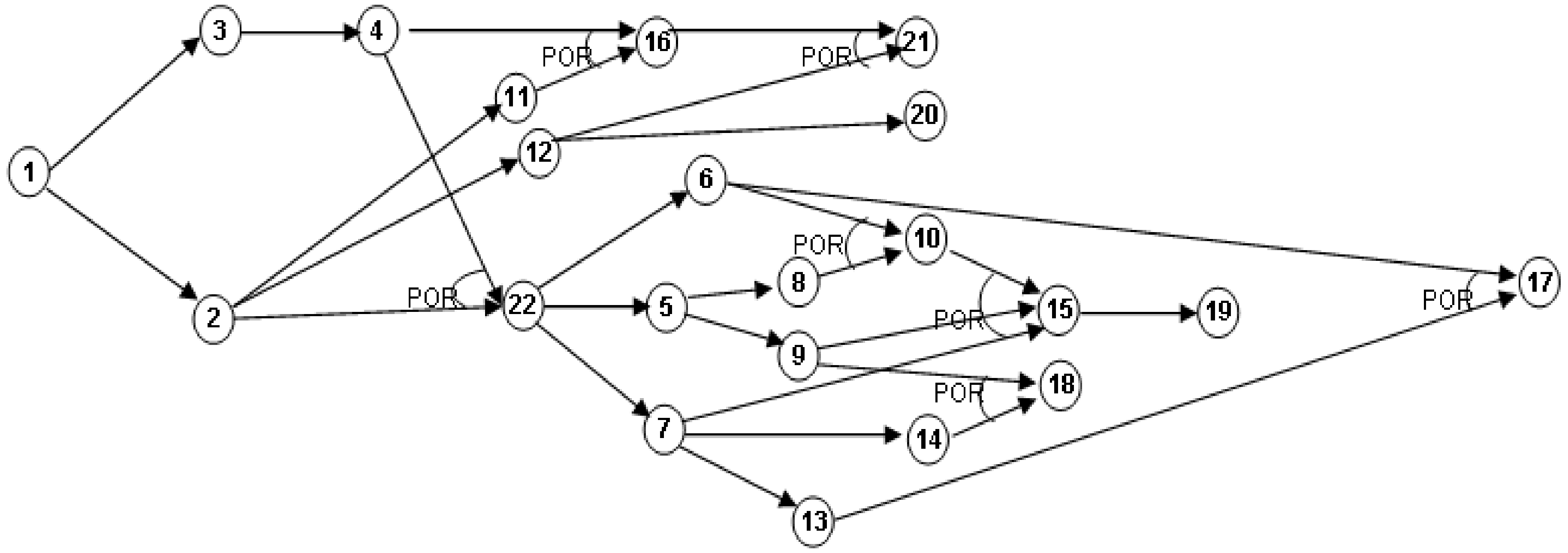


Figure 5.3: Precedence diagram of the Lambert's ball-point pen example (22 tasks)

2nd Network (34 tasks)

Lambert (1997) illustrates a 10-part radio example. The disassembly operations on a disassembly graph with 30 tasks are demonstrated in Figure 5.4.

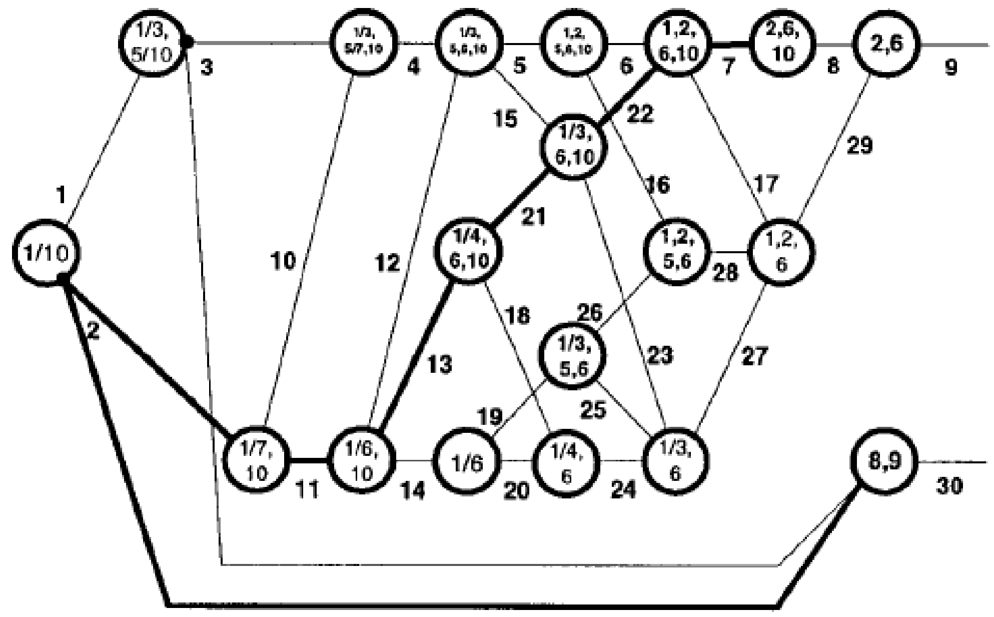


Figure 5.4: Disassembly graph of the 30 task 10 part radio example (Lambert, 1997)

The precedence diagram is formed by adding three dummy tasks (tasks 32_33_34) and is shown in the Figure 5.5.

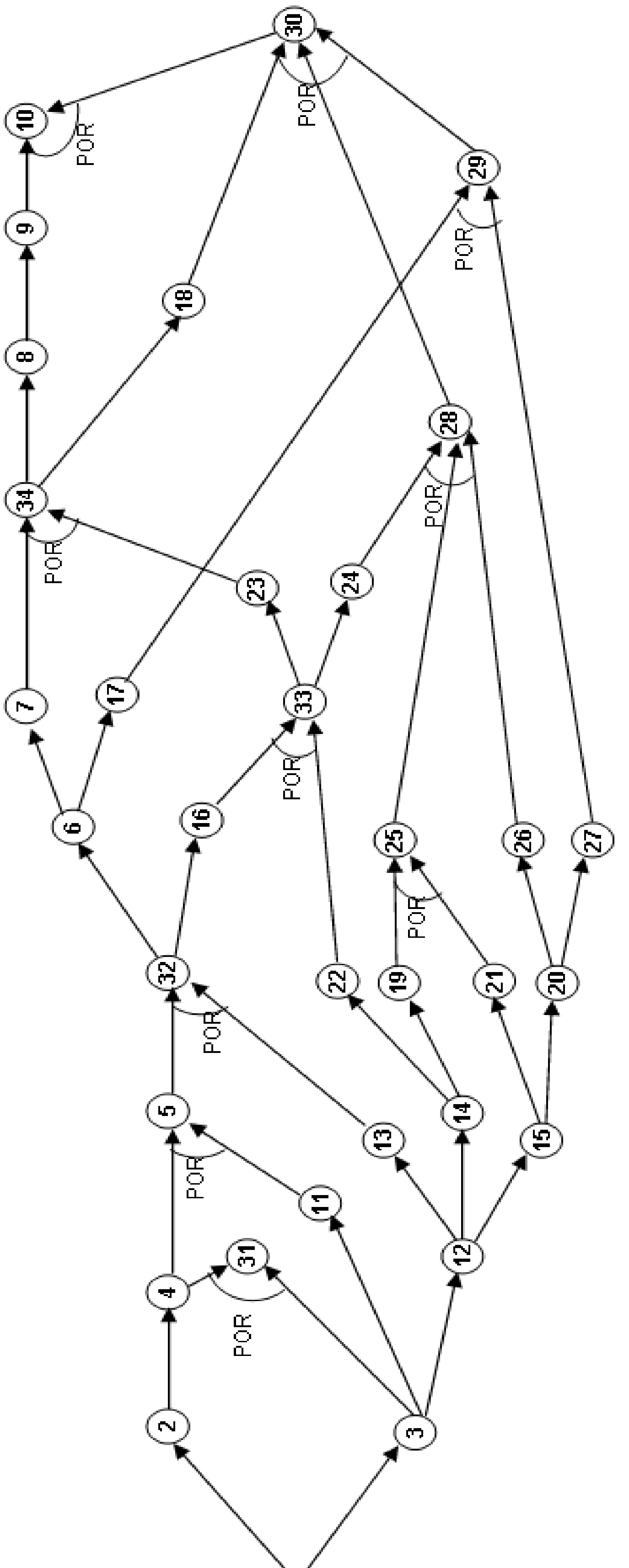


Figure 5.5: Precedence diagram of the Lambert's radio example (34 tasks)

3rd Network (47 tasks)

Our third network is formed by combining the second network (34 tasks) and a part of the first network (22 tasks_ used part is righthandside of task 22). Moreover some arcs are added to generate this 47 task network.

Precedence diagram of this network is shown in the Figure 5.6.

4th Network (60 tasks)

Our fourth network is formed by combining the third network (47 tasks) and a part of the first network (22 tasks_ used part is righthandside of task 22). More complicated arcs are added to generate this 60 task network.

Precedence diagram of this network is shown in the Figure 5.7.

5th Network (73 tasks)

Our fifth network is formed by combining the fourth network (60 tasks) and a part of the first network (22 tasks_ used part is righthandside of task 22). Moreover, some arcs are added to generate this 73 task network.

Precedence diagram of this network is shown in the Figure 5.8.

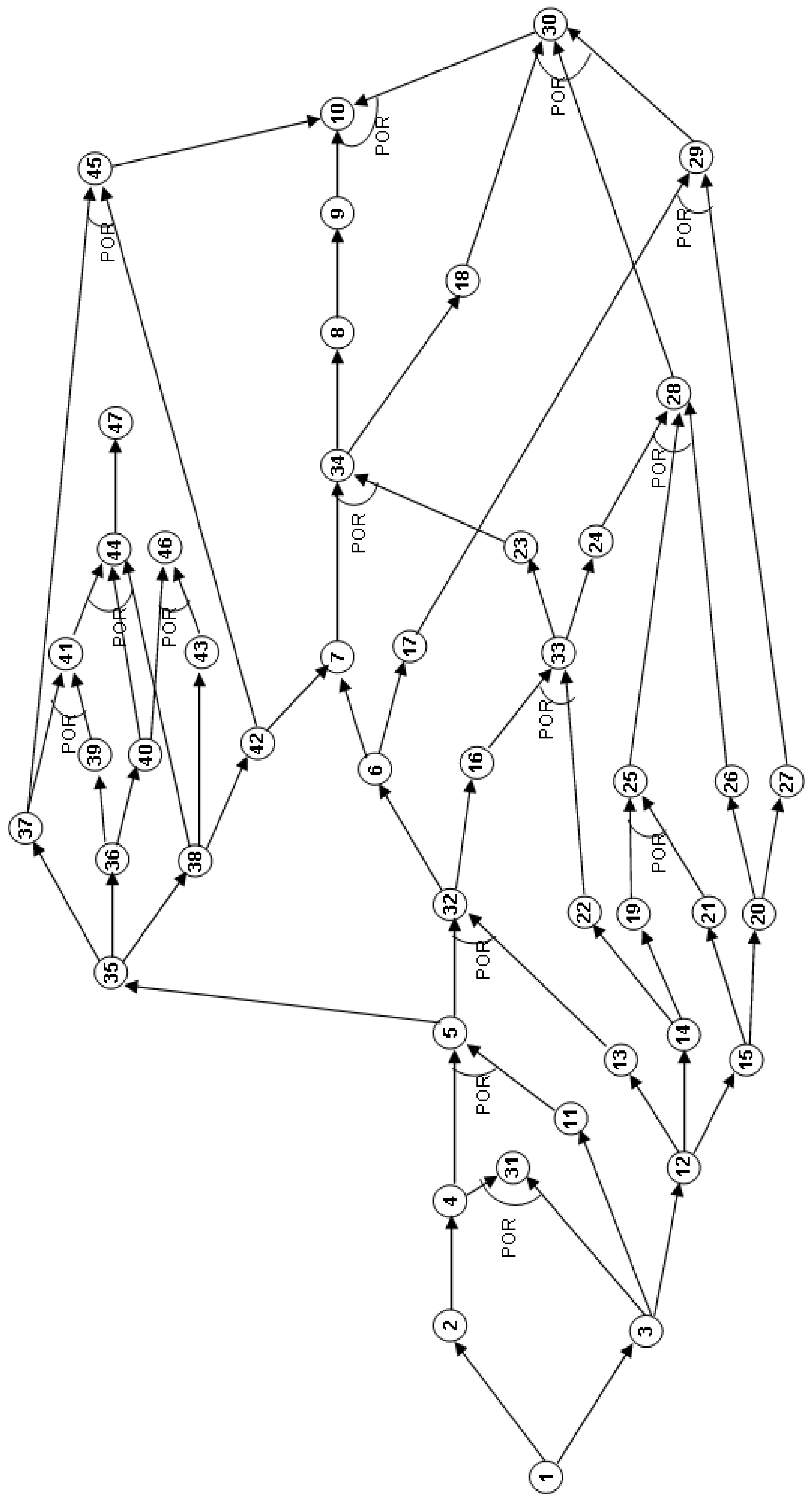


Figure 5.6: Precedence diagram of 47_task_Network

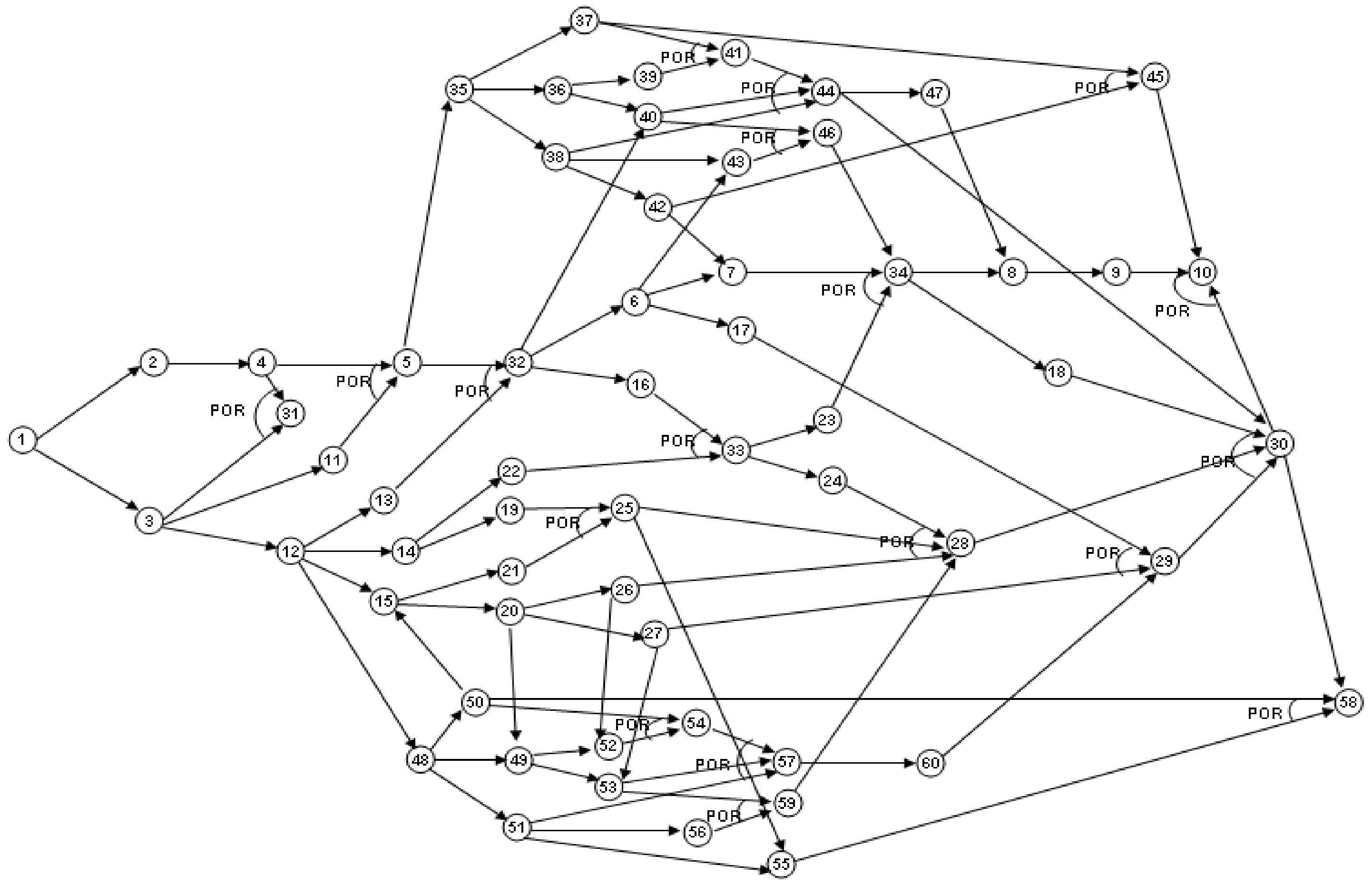


Figure 5.7: Precedence diagram of 60_task_Network

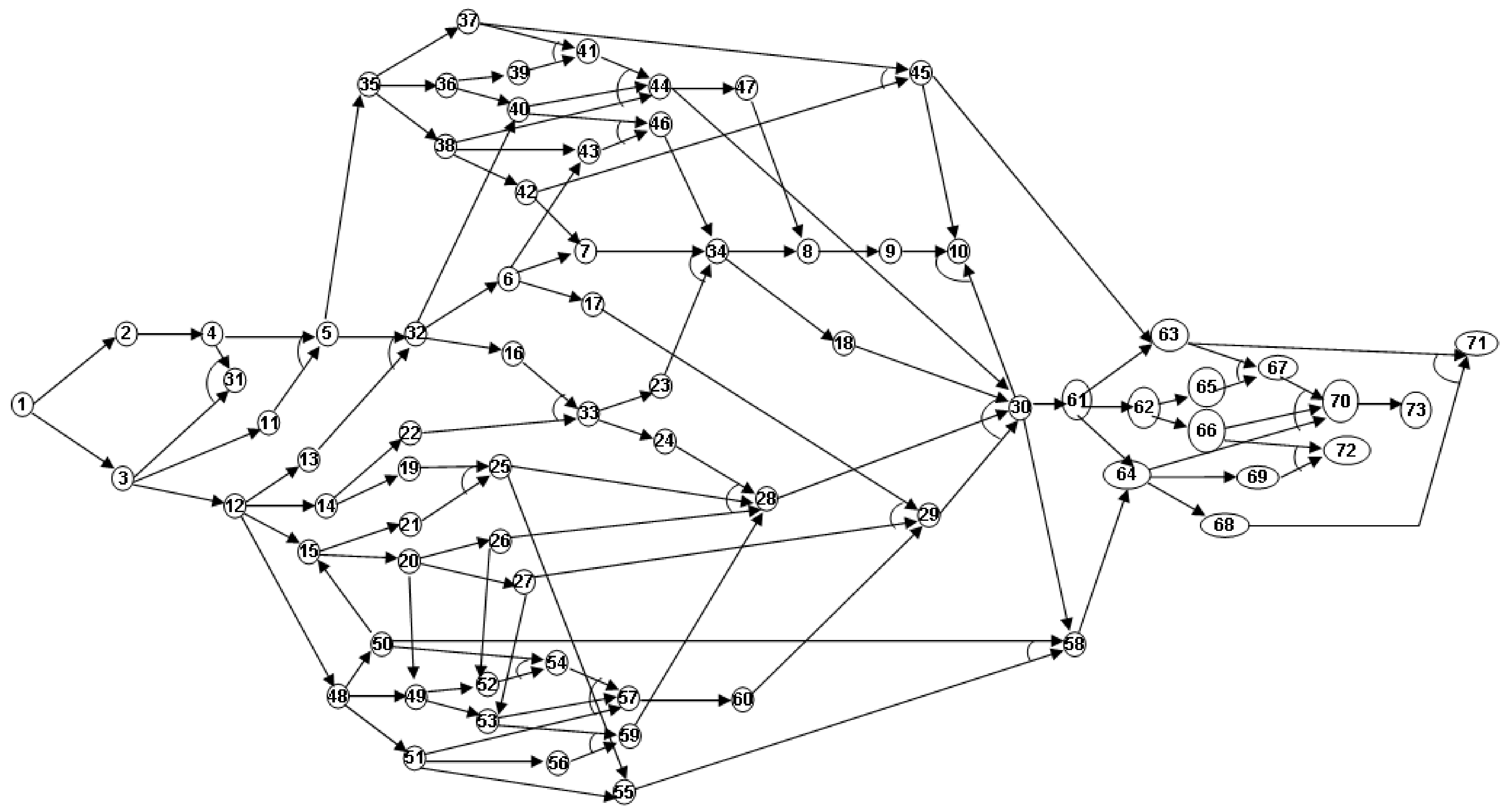


Figure 5.8: Precedence diagram of 73_task_Network

5.1.2. Parameter Generation

For each network we generate the *processing times* from a *discrete uniform distribution (1,20)*. After we generate the processing times, we use two sets for the cycle times.

Set 1 (C1) : Cycle time of an instance is set to $\left\lceil \frac{\sum t_i}{K} \right\rceil$ where *K* is the number of workstations.

Set 2 (C2) : Cycle time of an instance is set to $1,5 * \left\lceil \frac{\sum t_i}{K} \right\rceil$.

Our aim is to see the effect of the cycle times on the problem difficulty.

We use two values for the number of workstations *K*, for each network (*N*) and cycle time.

Set 1 (K1) : For small networks, *N*=20,34, 47 *K* is set to *2.*
For large networks, *N*=60,73 *K* is set to *4.*

Set 2 (K2) : For small networks, *K* is set to *4.*
For large networks, *K* is set to *8.*

A task receives a demand, hence called a part releasing task, according to a random process. We generate a random number between 0 and 1. If the generated number is below 0.3, we set its demand to zero. Hence we expect that about *%70* of all tasks are part releasing and attribute a demand and revenue for each part releasing task. We attribute a cost for all tasks.

For each value of *N*, *C*, and *K*, we use two distributions to generate the demands of the part releasing tasks.

Set 1 (D1) : Demand of a part releasing task is uniform *between 1 and 5.*
Set 2 (D2) : Demand of a part releasing task is uniform *between 5 and 10.*

Note that *D1* contains low demand and *D2* contains high demand problem instances.

For all instances, we set the number of used products, hence the number of periods, *T* to 80.

For all tasks, we generate the *costs* from a *discrete uniform distribution between 5 and 20.* For all part releasing tasks, we generate the *revenues* from a *discrete uniform distribution between 10 and 50*. We set the *net revenue* (profit) of each task as the *difference between its revenue and cost*.

We have 5, 2, 2 and 2 alternatives for each *N, K, C* and *D* respectively.
This leads to 3*2*2*2=24 combinations for small networks and 2*2*2*2=16 combinations for big networks. Thus a total of *40 combinations* are used.
For each combination, we generate and solve 10 problem instances. Hence we use *400 problem instances* in our experiments.

5.2. Performance Measures

In this section, we discuss the performance measures we use to evaluate the efficiency of our *Upper Bounds* and *Lower Bounds*.

We use the following performance measures for the *Upper Bounds*:

1. Deviation from the optimal (or best known) solution (average, maximum)
2. The solution time expressed as Central Processing Units (CPU) in seconds (average, maximum)
3. Total Number of fractional values (average, maximum)

We use the following performance measures for the *Lower Bound*:

1. Deviation from the optimal solution (for small networks) as a percentage of the optimal solution, Deviation from the Upper Bound (for large networks) (average, maximum)
2. The Central Processing Unit (CPU) time in seconds (average, maximum)

The optimal solutions are found by CPLEX 10.1. CPLEX is run for 3600 seconds. All experimentations are done in Intel Core2 Duo 2.00 GHz, 2 GB RAM. All algorithms are coded with Microsoft Visual C++ 2008.

5.3. Discussion on Experiments

We first investigate the performance of our upper bounding procedures. We report the average and maximum deviations of the upper bounds in Tables 5.1, 5.2 and 5.3, for networks 22, 34 and 47 tasks respectively. We calculate the deviation (DEV) as;

$$\%Dev = \left(\frac{UBi - OPT}{OPT}\right) \times 100 \quad \text{where}$$

OPT = Optimal objective function value, optimal total revenue
UBi = Total revenue returned by upper bound i.

We use the following abbreviations to state our problem combinations.

K1_C1_D1: K1 is the small number of workstations in each network, *C1* is the small cycle time and *D1* is the small demand type. All of them stems from *Set1* of their related category.
K2_C2_D2: K2 is the high number of workstations in each network, *C2* is the high cycle time (1,5*C1) and *D2* is the high demand type. All of them stems from *Set2* of their related category.

A total of 8 abbreviations are used for 8 combinations which are; *K1_C1_D1, K1_C1_D2, K1_C2_D1, K1_C2_D2, K2_C1_D1, K2_C1_D2, K2_C2_D1* and *K2_C2_D2.*

Table 5.1: Deviation of UB1 and UB2 from optimal and Number of fractions of UB1 and UB2 for N=22

	# of Variables		C1				C2				# of Optimal Solutions
			D1		D2		D1		D2		
			Avg	Max	Avg	Max	Avg	Max	Avg	Max	
UB1 _ K1	3,520	DEV	0.01	0.02	0.01	0.02	0.01	0.02	0.01	0.02	10
		# of frac.	722.40	1636.00	1120.70	1778.00	487.10	1356.00	470.60	1684.00	10
UB1 _ K2	7,040	DEV	0.01	0.03	0.01	0.03	0.01	0.03	0.01	0.03	10
		# of frac.	2437.50	3643.00	2497.40	3592.00	1628.10	3017.00	1900.40	3306.00	10
UB2 _ K1	3,520	DEV	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10
		# of frac.	65.30	160.00	67.00	160.00	0.00	0.00	0.00	0.00	10
UB2 _ K2	7,040	DEV	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	10
		# of frac.	368.40	570.00	376.00	596.00	197.80	432.00	180.30	412.00	10

Table 5.2: Deviation of UB1 and UB2 from optimal and Number of fractions of UB1 and UB2 for N=34

	# of Variables		C1				C2				# of Optimal Solutions
			D1		D2		D1		D2		
			Avg	Max	Avg	Max	Avg	Max	Avg	Max	
UB1 _ K1	5,440	DEV	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	10
		# of frac.	715.90	1950.00	1150.80	2251.00	786.00	2367.00	741.00	2210.00	10
UB1 _ K2	10,880	DEV	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	10
		# of frac.	3803.10	4864.00	4238.70	5136.00	3004.10	4307.00	3102.60	4386.00	10
UB2 _ K1	5,440	DEV	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10
		# of frac.	19.00	160.00	22.90	158.00	0.00	0.00	0.00	0.00	10
UB2 _ K2	10,880	DEV	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10
		# of frac.	525.20	870.00	517.70	855.00	271.10	438.00	259.10	446.00	10

Table 5.3: Deviation of UB1 and UB2 from optimal and Number of fractions of UB1 and UB2 for N=47

	# of Variables		C1				C2				# of Optimal Solutions
			D1		D2		D1		D2		
			Avg	Max	Avg	Max	Avg	Max	Avg	Max	
UB1 _ K1	7,520	DEV	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	10
		# of frac.	1184.00	4011.00	2509.40	3924.00	584.20	2403.00	1152.40	3391.00	10
UB1 _ K2	15,040	DEV	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	10
		# of frac.	5492.40	6865.00	5910.80	6966.00	4041.10	5888.00	4654.90	5946.00	10
UB2 _ K1	7,520	DEV	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10
		# of frac.	49.00	160.00	50.00	160.00	0.00	0.00	0.00	0.00	10
UB2 _ K2	15,040	DEV	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10
		# of frac.	570.30	970.00	592.00	982.00	370.40	800.00	362.20	815.00	10

Table 5.4: Number of fractions of UB1 and UB2 for N=60

	# of Variables		C1				C2				# of Optimal Solutions
			D1		D2		D1		D2		
			Avg	Max	Avg	Max	Avg	Max	Avg	Max	
UB1 _ K1	19,200	DEV	-	-	-	-	-	-	-	-	10
		# of frac.	8749.80	9579.00	8999.50	9791.00	6887.30	8344.00	7751.10	8935.00	10
UB1 _ K2	38,400	DEV	-	-	-	-	-	-	-	-	10
		# of frac.	13515.30	14742.00	13853.10	15153.00	11399.30	13268.00	12307.10	14280.00	10
UB2 _ K1	19,200	DEV	-	-	-	-	-	-	-	-	10
		# of frac.	746.00	1524.00	765.70	1546.00	418.30	1068.00	424.80	1068.00	10
UB2 _ K2	38,400	DEV	-	-	-	-	-	-	-	-	10
		# of frac.	1778.70	2203.00	1725.20	2116.00	1192.10	1523.00	1179.20	1434.00	10

Table 5.5: Number of fractions of UB1 and UB2 for N=73

	# of Variables		C1				C2				# of Optimal Solutions
			D1		D2		D1		D2		
			Avg	Max	Avg	Max	Avg	Max	Avg	Max	
UB1 _ K1	23,360	DEV	-	-	-	-	-	-	-	-	10
		# of frac.	10876.30	11804.00	11249.80	11974.00	9137.30	10462.00	9716.30	10633.00	10
UB1 _ K2	46,720	DEV	-	-	-	-	-	-	-	-	10
		# of frac.	16730.70	18440.00	17291.30	18654.00	14045.50	15689.00	15019.80	16147.00	10
UB2 _ K1	23,360	DEV	-	-	-	-	-	-	-	-	10
		# of frac.	657.50	922.00	649.90	859.00	460.80	1027.00	462.40	1045.00	10
UB2 _ K2	46,720	DEV	-	-	-	-	-	-	-	-	10
		# of frac.	1789.10	2326.00	1769.00	2244.00	1197.10	1931.00	1201.80	2008.00	10

As can be observed from the tables the deviations are consistently low over all problem combinations. For UB_1 the average deviations are *below %1* when *N*=22, *34 and 47* respectively. The maximum deviations are also shown in Tables 5.1, 5.2 and 5.3 which are *below %3* for all combinations.

Note that the deviations do not deteriorate with an increase in *N* value. However the deviations increase by the increase in *K* value due to the inflation of the number of variables. When *N* and *K* values are fixed, demand (D) and cycle time (C) values do not effect deviations remarkably.

UB_2 produces slightly smaller deviations due to the power of our cuts. For example the maximum deviation *for UB_1 is %3,* whereas it is *nearly zero for UB_2* in the combination of *K1_C1_D1* for *N*=22.

We also give the number of fractional variables in Tables 5.1, 5.2 and 5.3. The numbers of the fractional variables produced by Pure LP Relaxation, *UB_1 s*, are quite high. For example when *N*=22, for combination *K2_C2_D2*, i.e., 4 workstations, high demand and high cycle time case, 3306 out of 7040 variables are found to be fractional, at worst case. This value reduces to 412 when cuts are incorporated. On average, the cuts reduce the number of fractional variables from 1900.4 to 180.3. The number of fractional variables is generally affected by *N* values. For example, from Tables 5.1, 5.2, 5.3, 5.4 and 5.5, it can be seen that for combination *K2_C2_D2,* increasing *N* increases the number of fractional variables of UB_1,on average. Note that average number of fractional variables are 1900.4, 3102.6, 4654.9, 12307.1 and 15019.8 for *N = 22, 34, 47, 60 and 73* respectively.

We also observe from Tables 5.1 through 5.5 that, for fixed N, an increase in the number of workstations increases the number of fractional variables. For example for N=22 and C1_D1 combination, the number of fractional variables is 722.4 when K=2 (K1) and 2437.5 when K= 4 (K2), on average. This is due to the fact that more workstations lead to higher number of splits, hence more fractional variables.

For fixed *N* and *K*, we observe that increasing *C* reduces the number of fractional variables with a few exceptions. For example when *N=47, K=2*, and D1 is used for combination, the average number of fractional variables are 1184 and 584.2 when *C* is *low* and *high* respectively. This is due to the fact that increasing *C* gives more room to the complete task assignments, hence reducing the number of fractional variables.

In our experiments we do not observe a significant effect of the demand figures on the number of fractional variables.

We measure the solution times in Central Processing Unit (CPU) seconds. The average and maximum *CPU times for our three bounds* and *for the MILP* are given in Tables 5.6 through 5.10.

Table 5.6: CPU Times of Upper Bounds and MILP for N=22

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB1	0.33	0.50	0.42	0.54	0.22	0.28	0.23	0.34
	UB2	0.09	0.10	0.11	0.14	0.07	0.09	0.07	0.08
	UB3	0.05	0.09	0.06	0.08	0.05	0.06	0.06	0.08
	MILP	3.83	10.67	10.65	73.21	1.76	2.00	1.77	2.15
K2	UB1	0.70	1.25	0.84	1.09	0.47	0.61	0.59	0.73
	UB2	0.41	0.67	0.40	0.65	0.23	0.35	0.25	0.36
	UB3	0.07	0.10	0.09	0.11	0.06	0.07	0.08	0.10
	MILP	1078.32	3599.85	796.07	3600.62	363.90	3599.87	365.29	3600.30

* Tabulated data is obtained out of 10 optimal instances.

Table 5.7: CPU Times of Upper Bounds and MILP for N=34

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB1	0.57	0.86	0.78	0.92	0.36	0.44	0.43	0.53
	UB2	0.11	0.12	0.11	0.13	0.09	0.11	0.08	0.11
	UB3	0.06	0.08	0.07	0.10	0.06	0.08	0.06	0.07
	MILP	4.37	6.04	5.93	8.15	3.22	4.32	3.41	4.44
K2	UB1	1.27	1.95	1.67	2.21	0.91	1.14	1.18	1.34
	UB2	0.44	0.53	0.46	0.58	0.32	0.39	0.31	0.38
	UB3	0.08	0.10	0.11	0.12	0.08	0.09	0.09	0.11
	MILP	1454.94	3600.34	939.75	3417.72	373.65	3599.81	19.76	27.86

* Tabulated data is obtained out of 10 optimal instances.

Table 5.8: CPU Times of Upper Bounds and MILP for N=47

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB1	1.05	1.36	1.42	1.63	0.48	0.69	0.65	0.87
	UB2	0.16	0.18	0.14	0.16	0.11	0.12	0.12	0.13
	UB3	0.07	0.11	0.08	0.12	0.06	0.07	0.07	0.09
	MILP	22.77	118.85	14.38	64.45	5.65	6.88	5.62	7.02
K2	UB1	2.46	3.59	2.94	3.51	1.52	1.90	2.17	2.86
	UB2	0.82	1.17	0.79	1.05	0.55	0.83	0.56	0.84
	UB3	0.09	0.11	0.16	0.20	0.09	0.10	0.12	0.17
	MILP	784.49	3600.90	759.57	3600.60	376.38	3601.31	385.74	3599.83

* Tabulated data is obtained out of 10 optimal instances.

Table 5.9: CPU Times of Upper Bounds for N=60

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB1	4.27	5.03	4.89	5.82	2.47	2.94	3.18	3.66
	UB2	1.21	1.69	1.26	1.98	0.72	1.12	0.72	1.21
	UB3	0.10	0.12	0.16	0.23	0.08	0.10	0.11	0.15
K2	UB1	13.87	18.60	19.59	25.51	6.82	8.67	9.44	11.55
	UB2	5.36	6.79	4.97	6.76	2.52	3.13	2.67	3.44
	UB3	0.17	0.21	0.38	0.54	0.15	0.18	0.24	0.33

* Tabulated data is obtained out of 10 optimal instances.

Table 5.10: CPU Times of Upper Bounds for N=73

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB1	6.27	8.80	7.59	9.42	3.85	4.37	4.52	5.21
	UB2	1.57	2.23	1.49	2.51	0.79	1.13	0.83	1.14
	UB3	0.11	0.14	0.17	0.22	0.11	0.15	0.13	0.16
K2	UB1	19.99	23.34	32.14	42.58	9.35	10.88	13.59	17.40
	UB2	5.83	8.80	6.92	10.83	3.63	5.06	3.31	4.16
	UB3	0.21	0.31	0.56	0.99	0.15	0.17	0.29	0.45

* Tabulated data is obtained out of 10 optimal instances.

As can be observed from the tables, the upper bounds are produced in very small times. Compared to UB_1, UB_2 runs in smaller times due to the efficiency of the cuts, i.e., their power in reducing the solution space. Moreover, compared to UB_2, UB_3 runs in smaller times as relatively fewer periods, hence fewer variables are used by the LP Relaxations. For example for *N*=22 and *K2_C2_D2* combination, the CPU time decreases from 0.59 to 0.25 seconds by adding the cuts and the CPU time decreases from 0.25 to 0.08 seconds by using up to 10 periods instead of 80.

The CPU times spent by the upper bounds, increase with an increase in problem size parameters, *N* and *K*. This is due to the increase in the dimensions of the linear programs. As *N* increases from 22 to 73 (with fixed *K* value), for example for *K1_C2_D2* combination, the average CPU times increase from 0.23, 0.07, 0.06 to 4.52, 0.83, 0.13 for UB_1, UB_2 and UB_3 respectively. For *N*=34 and *C2_D2* combination, as *K* increases, the average CPU times increase from 0.43, 0.08, 0.06 seconds to 1.18, 0.31, 0.09 seconds for UB_1, UB_2 and UB_3 respectively.

When *N, K and D* values are fixed, an increase in the *C* value (from *C1 to C2*) decreases the CPU time. This is due to the fact that for large C more tasks find place for any workstation, and hence assignment decisions are given easier by linear programs. For example for *N*=34 and *K2_D2* combination, increasing *C* from *C1 to C2*, decreases the average CPU times from 1.67, 0.46, 0.11 seconds to 1.18, 0.31, 0.09 seconds for UB_1, UB_2 and UB_3, respectively.

When the other parameters are fixed, the increase in *D* value (from *D1 to D2*) increases the CPU times slightly. This is because when the *D* value increases, the number of periods with different assignments increases. For example for *N*=34 and *C1_K2* combination, increasing *D* from *D1 to D2*, increases the average CPU times from 1.27, 0.44, 0.08 seconds to 1.67, 0.46, 0.11 seconds for UB_1, UB_2 and UB_3 respectively.

The effects of *K* is much dominant for the MILP, due to the inflation of the binary decision variables. For *fixed N,* an increase in *K* value increases the CPU times increases remarkably. For example for *N*=22 and *C1_D1* combination, an increase of *K* from *K1 to K2* increases the CPU time from 3.83 seconds to 1078.32 seconds. However the effect of *N* is not significant as that of *K*. For example, for *K=2* when *N* increases from 22 to 34 for *C1_D1* combination, the CPU time increases from 3.83 seconds to 4.37 seconds.

As can be seen from Tables 5.6 through 5.10, when *N* and *K* values are fixed, an increase in the *C* value decreases the CPU times remarkably. This is due to the fact that a workstation has more to accommodate many tasks, thereby leading to easier decisions. However the *D* values do not have a consistent affect on the solution time of the MILP.

As mentioned, the speed performance of UB_3 can be attributed to the small number of periods it uses. We tabulate the number of periods that we start with, i.e., *LB(T)* and we find the first feasible solution , i.e., $UB_3(T)$ in Tables 5.11 through 5.15 for *N*=22, 34, 47, 60 and 73, respectively.

Table 5.11: Number of Periods used by UB3 (out of 80) for N=22

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB3_T	4.30	5.00	10.00	11.00	4.50	5.00	10.20	11.00
	LB_(T)	2.30	3.00	6.00	7.00	1.50	2.00	4.20	5.00
K2	UB3_T	4.10	5.00	10.80	12.00	4.30	5.00	9.70	10.00
	LB_(T)	2.10	3.00	5.80	7.00	1.30	2.00	3.70	4.00

* Tabulated data is obtained out of 10 optimal instances.

Table 5.12: Number of Periods used by UB3 (out of 80) for N=34

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB3_T	4.10	5.00	9.00	9.00	4.90	5.00	9.00	9.00
	LB_(T)	2.10	3.00	6.00	6.00	1.90	2.00	4.00	4.00
K2	UB3_T	4.00	4.00	9.90	10.00	4.10	5.00	9.00	9.00
	LB_(T)	2.00	2.00	5.90	6.00	1.10	2.00	4.00	4.00

* Tabulated data is obtained out of 10 optimal instances.

Table 5.13: Number of Periods used by UB3 (out of 80) for N=47

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB3_T	4.50	5.00	10.30	11.00	4.80	5.00	10.20	11.00
	LB_(T)	2.50	3.00	6.30	7.00	1.80	2.00	4.20	5.00
K2	UB3_T	4.30	5.00	10.10	11.00	4.40	5.00	10.00	11.00
	LB_(T)	2.30	3.00	6.10	7.00	1.40	2.00	4.00	5.00

* Tabulated data is obtained out of 10 optimal instances.

Table 5.14: Number of Periods used by UB3 (out of 80) for N=60

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB3_T	4.20	5.00	9.30	10.00	4.40	5.00	9.00	9.00
	LB_(T)	2.20	3.00	6.30	7.00	1.40	2.00	4.00	4.00
K2	UB3_T	4.00	4.00	9.10	10.00	4.10	5.00	9.00	9.00
	LB_(T)	2.00	2.00	6.10	7.00	1.10	2.00	4.00	4.00

* Tabulated data is obtained out of 10 optimal instances.

Table 5.15: Number of Periods used by UB3 (out of 80) for N=73

		C1				C2			
		D1		D2		D1		D2	
		Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	UB3_T	4.20	5.00	9.30	10.00	4.70	5.00	9.10	10.00
	LB_(T)	2.20	3.00	6.30	7.00	1.70	2.00	4.10	5.00
K2	UB3_T	4.10	5.00	9.20	10.00	4.20	5.00	9.10	10.00
	LB_(T)	2.10	3.00	6.20	7.00	1.20	2.00	4.10	5.00

* Tabulated data is obtained out of 10 optimal instances.

As can be observed from the tables, the *LB(T)* and *$UB_3(T)$* values are close, hence *LB(T)* performs quite satisfactory. For example for *N*=22 and *K1_C1_D1* combination, the average *LB(T)* and *$UB_3(T)$* values are 2.30 and 4.30 respectively. We also find that the *$UB_3(T)$* values are small and are not sensitive to the problem size. For example for *K1_C1_D1* combination, the average *$UB_3(T)$* values are 4.30 and 4.20 for *N*=22 and 73 respectively.

As mentioned before, small numbers periods produces few decision variables, hence improves the speed performance of the upper bounds considerably. Note that we obtain the same objective function values from UB_2 and UB_3, however at considerably different speeds.

We finally investigate the performances of our lower bound. We measure the deviations for the small-sized problems using the optimal solutions. For large-sized problems, we use *UB_2,* equivalently *UB_3*, to find the deviation as the optimal solutions are not available. Particularly when *N=22,34 and 47*, we use;

$$\%Dev = \left(\frac{OPT - LB}{OPT} \right) \times 100$$

When *N*=60 and 73, we do not have optimal solutions on hand. We use UB_2 as an estimator for the optimal objective function value and find the deviation as;

$$\%Dev = \left(\frac{UB_2 - LB}{UB_2} \right) \times 100$$

We report the average and maximum deviations of the lower bound in Tables 5.16 through 5.20 for *N=22,34,47,60 and 73*, respectively.

Table 5.16: Deviation of LB from Optimal and CPU Times of LB for N=22

	# of Variables		C1				C2			
			D1		D2		D1		D2	
			Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	3,520	DEV	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00
		CPU	0.06	0.13	0.08	0.12	0.07	0.09	0.07	0.08
K2	7,040	DEV	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.00
		CPU	0.10	0.15	0.12	0.16	0.08	0.10	0.10	0.14

* Tabulated data is obtained out of 10 optimal instances.

Table 5.17: Deviation of LB from Optimal and CPU Times of LB for N=34

	# of Variables		C1				C2			
			D1		D2		D1		D2	
			Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	5,440	DEV	0.00	0.00	0.01	0.02	0.00	0.00	0.00	0.00
		CPU	0.05	0.08	0.08	0.08	0.09	0.11	0.08	0.11
K2	10,880	DEV	0.01	0.02	0.01	0.02	0.01	0.02	0.01	0.01
		CPU	0.03	0.04	0.13	0.15	0.10	0.12	0.11	0.14

* Tabulated data is obtained out of 10 optimal instances.

Table 5.18: Deviation of LB from Optimal and CPU Times of LB for N=47

	# of Variables		C1				C2			
			D1		D2		D1		D2	
			Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	7,520	DEV	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00
		CPU	0.09	0.15	0.10	0.14	0.11	0.12	0.12	0.13
K2	15,040	DEV	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01
		CPU	0.12	0.14	0.19	0.23	0.12	0.14	0.15	0.21

* Tabulated data is obtained out of 10 optimal instances.

Table 5.19: Deviation of LB from UB2 and CPU Times of LB for N=60

	# of Variables		C1				C2			
			D1		D2		D1		D2	
			Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	19,200	DEV	0.00	0.01	0.01	0.01	0.00	0.01	0.00	0.01
		CPU	0.12	0.14	0.18	0.25	0.10	0.13	0.13	0.18
K2	38,400	DEV	0.00	0.01	0.01	0.02	0.01	0.01	0.01	0.01
		CPU	0.20	0.24	0.41	0.59	0.18	0.23	0.28	0.38

Table 5.20: Deviation of LB from UB2 and CPU Times of LB for N=73

	# of Variables		C1				C2			
			D1		D2		D1		D2	
			Avg	Max	Avg	Max	Avg	Max	Avg	Max
K1	23,360	DEV	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.01
		CPU	0.14	0.17	0.18	0.25	0.14	0.20	0.16	0.21
K2	46,720	DEV	0.01	0.01	0.01	0.02	0.01	0.01	0.01	0.02
		CPU	0.26	0.34	0.61	1.04	0.19	0.22	0.32	0.49

Note from the tables that the lower bound solutions are quite satisfactory over all problem combinations. The solutions have small deviations and are obtained in negligible CPU times. The performances do not deteriorate with an increase in the problem size parameter *N*; however the deviations slightly increase by an increase in the *K* value. For example for *N*=60 and C1_D2 combination, the average deviations of *K1* and *K2* are *%1*; whereas the maximum deviations are *%1 and %2* for *K1* and *K2*, respectively. We could not observe notable effects of *C* and *D* values on the problem difficulty.

The *CPU times* of the lower bounds increase with an increase in problem size parameters, *N and K* with a few exceptions. This is due to the fact that the linear programs and mixed integer linear programs have higher number of decision variables for higher values of *N and K.* For example for *K1_C2_D2* combination, as *N* increases from 60 to 73, the average CPU times increase from 0.13 seconds to 0.16 seconds. For *N*=47 and *C2_D2* combination, increasing *K* from *2* to *4*, increases the average CPU times from 0.12 seconds to 0.15 seconds.

We could not observe any significant effect of *C* value on the CPU times.

For fixed *N, K and C* values, increasing the *D* value increases the CPU times slightly. This increase can be attributed to the increase in the value of *LB(T),* hence the number of decision variables that explain the periods. For example for *N=60* and *C1_K2* combination, increasing *D* from *D1 to D2* , increases the average CPU time from 0.20 seconds to 0.41 seconds.

CHAPTER 6

CONCLUSIONS AND FURTHER RESEARCH DIRECTIONS

In this study, we consider disassembly systems that have gained significant importance in recent years. This heightened importance stems from the recognition of environmental issues and the advances in manufacturing technologies. In this study, we consider an operational level problem in disassembly systems, the so called disassembly line balancing problem.

Our problem assumes that the line is already configured with defined workstations. The units of the product to be disassembled are identical and they deliver parts with defined demand and revenue. The tasks that release a part or are required for further releases have defined costs. Our aim is to assign the tasks to the workstations so that the net revenue is maximized.

We develop a Mixed-Integer Linear Programming (MILP) model that could solve the problems with up to 50 tasks. For larger sized instances, we propose upper and lower bounds. The bounds are motivated by our findings from the highly satisfactory behavior of the Linear Programming Relaxations (LPR). We strengthen LPR by imposing the properties that are satisfied by the MILP but not LPR. Our lower bound fixes the integer variables of the optimal LPR solution and solves the remaining problem to optimality by MILP.

Our experimental results have revealed that our bounding mechanisms produce high quality solutions very quickly. For our maximum trial size of 75 tasks there is a gap of less than 5 percent between our lower and upper bounds.

To the best of our knowledge, our study is the first attempt to solve the disassembly line balancing problem with a fixed number of workstations and a finite supply of disassembly products. The extensions of our study may include the following issues:

- Incorporation of SOR (Successor OR) type precedence relations
- Considering nonidentical products, i.e., each unit of the disassembly product may include different parts
- Incorporating the stochastic nature of the outcome, i.e., some parts may turn out to be defective, or be damaged during disassembly.
- Treating the number of workstations as a decision variable, hence considering the design version of the disassembly line balancing problem.

REFERENCES

Altekin, F.T., Kandiller L. and Özdemirel N.E., (2008), "Profit Oriented Disassembly Line Balancing", *Journal of Production Research,* 46, 2675-2693.

Baybars, I. (1986), "A Survey of Exact Algoritms for the Simple Assembly Line Balancing Problem", *Management Science* 32, 909-932.

Brennan, L., Gupta, S.M., and Taleb, K.N., (1994), "Operations Planning Issues in an Assembly/Disassembly Environment", *International Journal of Operations and Production Management,* 14, 57-67.

Güngör, A., and Gupta, S.M., (2001), "A solution approach to the disassembly line balancing problem in the presence of task failures", *International Journal of Production Research*, 39, 1427-1467.

Güngör, A., and Gupta, S.M., (2002), "Disassembly line in product recovery", *International Journal of Production Research*, 40, 2569-2589.

Lambert, A. J. D., (1997), "Optimal Disassembly of Complex Products", *International Journal of Production Research*, 35, 2509-2523.

Lambert, A. J. D., (1999), "Linear programming in disassembly/clustering sequence generation", *Computers and Industrial Engineering*, 36, 723-738.

Lambert, A. J. D., (2002), "Determining optimum disassembly sequences in electronic equipment", *Computers and Industrial Engineering*, 43, 553-575.

Martello, S., and Toth, P., (1990). *"Knapsack Problems: Algorithms and Computer Implementations*", John Wiley & Sons, ISBN: 0-471-92420-2.

McGovern, S.M., and Gupta, S.M., (2007), "A balancing method and genetic algorithm for disassembly line balancing", *European Journal of Operational Research*, 179, 692-708.

Scholl, A., (1999), *"Balancing and sequencing of disassembly lines*", 2nd ed., Physica Heidelberg.