DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

RESOURCE CONSTRAINED PARALLEL MACHINE SCHEDULING PROBLEMS WITH MACHINE ELIGIBILITY RESTRICTIONS: MATHEMATICAL AND CONSTRAINT PROGRAMMING BASED APPROACHES

by
Emrah B. EDİS

December, 2009
İZMİR

RESOURCE CONSTRAINED PARALLEL MACHINE SCHEDULING PROBLEMS WITH MACHINE ELIGIBILITY RESTRICTIONS: MATHEMATICAL AND CONSTRAINT PROGRAMMING BASED APPROACHES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Industrial Engineering, Industrial Engineering Program

by
Emrah B. EDİS

December, 2009
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"RESOURCE CONSTRAINED PARALLEL MACHINE SCHEDULING PROBLEMS WITH MACHINE ELIGIBILITY RESTRICTIONS:MATHEMATICAL AND CONSTRAINT PROGRAMMING BASED APPROACHES"** completed by **EMRAH B. EDİS** under supervision of **PROF. DR. HASAN ESKİ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Hasan ESKİ

Supervisor

Prof. Dr. Tatyana YAKHNO	Asst. Prof. Dr. Şeyda A. TOPALOĞLU
Thesis Committee Member	Thesis Committee Member
Prof. Dr. A. İrem ÖZKARAHAN	Assoc. Prof. Dr. Ceyda OĞUZ
Second Supervisor	Examining Committee Member
Prof. Dr. Urfat NURİYEV	Asst. Prof. Dr. Bilge BİLGEN
Examining Committee Member	Examining Committee Member

Prof. Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest appreciation to my advisor, Prof. Dr. İrem ÖZKARAHAN for her continuous support, guidance and confidence in me throughout my PhD studies. Without her motivation, sincere support, valuable insights and advices, this dissertation could not have been completed. Also, I would like to thank to my other advisor, Prof. Dr. Hasan ESKİ who provided a continuous support, motivation and sincere interest throughout the progress of this dissertation.

I would like to acknowledge my PhD committee members, Asst. Prof. Dr. Şeyda TOPALOĞLU and Prof. Dr. Tatyana YAKHNO, for their constructive criticism and valuable comments during the progress of this dissertation. I would also like to extend my gratitude to Assoc. Prof. Dr. Ceyda OĞUZ for the short time she took to review my dissertation and provide valuable comments and suggestions that improved the quality and presentation of this dissertation.

I take this opportunity to thank all the professors and colleagues in the Department of Industrial Engineering at Dokuz Eylul University for their encouragement and friendship. Special thanks go to Pınar MIZRAK ÖZFIRAT, Ceyhun ARAZ, Kemal ÖZFIRAT, Özlem UZUN ARAZ and Gökalp YILDIZ for their help, endless support and encouragement.

I would also like to thank to Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing me scholarship during this study.

Last but not the least; I am deeply thankful to my family. Their sincere support and prayers were always with me. Finally, words are insufficient to express my deepest gratitude to my wife Rahime. Without her love, endless encouragement, understanding and assistance, this dissertation would not have been completed.

Emrah B. EDİS

RESOURCE CONSTRAINED PARALLEL MACHINE SCHEDULING PROBLEMS WITH MACHINE ELIGIBILITY RESTRICTIONS: MATHEMATICAL AND CONSTRAINT PROGRAMMING BASED APPROACHES

ABSTRACT

The research in this dissertation is motivated by a real-world scheduling problem in the injection molding department of an electrical appliance company and investigates three resource-constrained parallel machine scheduling problems with machine eligibility restrictions. All the problems consider one additional resource type (i.e., operator) with arbitrary resource size and up to two units of resource requirements.

The first problem assumes that processing times of all jobs are equal and aims to minimize total flow time. For this problem, two heuristic algorithms are proposed. The first one is a Lagrangian-based solution approach embedded into a subgradient optimization procedure to obtain tight lower bounds and near-optimal solutions. The second one is a problem specific heuristic. The performances of the proposed algorithms are evaluated by means of randomly generated test instances with different problem parameters.

The second problem allows arbitrary processing times and aims to minimize makespan. For this problem, three optimization models, namely, integer programming (IP), constraint programming (CP), and combined IP/CP models, are developed. Four different CP search algorithms have been evaluated and the best one is embedded into the CP and IP/CP combined models. The proposed models are then tested through medium size test problems and the efficiency of the proposed IP/CP combined model is demonstrated.

The last problem considers the real case with 36 machines and real die-machine compatibility data. For this problem, IP/IP and IP/CP iterative approaches which

partition the entire problem into loading and scheduling phases are proposed. Both approaches have a common loading phase where an IP model assigns the jobs to the machines. Subsequently, in the scheduling phase, two alternative models, namely, IP and CP are developed to construct the final schedule. The proposed approaches are evaluated by the test problems generated on real data, and the efficiency of IP/CP iterative approach is established.

Keywords: parallel machine scheduling, additional resources, machine eligibility restrictions, Lagrangian relaxation, integer programming, constraint programming.

MAKİNE ELVERİŞLİLİĞİ SINIRLAMALARI ALTINDAKİ KAYNAK KISITLI PARALEL MAKİNE ÇİZELGELEME PROBLEMLERİ: MATEMATİKSEL VE KISIT PROGRAMLAMA TABANLI YAKLAŞIMLAR

ÖZ

Bu tezdeki araştırmada, elektrik malzemeleri üreten bir firmanın plastik-enjeksiyon bölümündeki gerçek çizelgeleme probleminden motive olunmuş ve iş-makine elverişliliği altındaki ek kaynak kısıtlı üç adet paralel makine çizelgeleme problemi analiz edilmiştir. İncelenen tüm problemler, mevcut sayısı rastgele alınabilecek ancak gereksinimi en fazla iki adet olabilecek tek tip bir ek kaynağı (örn. operatör) dikkate almaktadır.

Ele alınan ilk problem tüm işlerin işlem sürelerini eşit kabul etmekte ve toplam akış zamanını en küçüklemeyi amaçlamaktadır. Bu problem için iki sezgisel yaklaşım önerilmiştir. İlk yöntem, sıkı alt sınır değerleri ve en iyi sonuca yakın üst sınır değerleri elde etmek üzere alt-gradyan eniyileme prosedürüne iliştirilmiş Lagrange-tabanlı bir çözüm yaklaşımıdır. İkinci yöntem ise probleme özgü sezgisel bir yaklaşımdır. Farklı problem parametreleri dikkate alınarak türetilen test problemleri üzerinde modellerin performansları değerlendirilmiştir.

Ele alınan ikinci problem, işlem sürelerinin keyfi olarak seçilebilmesine izin vermekte ve işlerin en son bitiş süresini (makespan) en küçüklemeyi amaçlamaktadır. Bu problem için, tamsayılı programlama (TP), kısıt programlama (KP) ve bütünleşik TP/KP olmak üzere üç farklı eniyileme modeli geliştirilmiştir. Dört farklı KP arama algoritması test edilmiş ve içlerinden en iyisi KP ve TP/KP bütünleşik modeline eklenmiştir. Önerilen modeller orta büyüklükteki test problemlerine uygulanmış ve TP/KP bütünleşik modelinin etkinliği gösterilmiştir.

Ele alınan son problem 36 makineden oluşan ve gerçek kalıp-makine elverişlilik verisini içeren çizelgeleme problemini ele almaktadır. Bu problem için, problemi yükleme ve çizelgeleme alt problemlerine ayıran TP/TP ve TP/KP ardışıksal

yaklaşımları önerilmiştir. Her iki yaklaşım, bir TP modelinin işleri makinelere atadığı ortak bir yükleme aşamasına sahiptir. Çizelgeleme aşamasında ise son çizelgeyi oluşturmak üzere TP ve KP olarak iki farklı model önerilmiştir. Gerçek verilere dayalı olarak türetilen test problemleri üzerindeki değerlendirmeler, TP/KP ardışıksal yaklaşımının etkinliğini ortaya koymuştur.

Anahtar Sözcükler: paralel makine çizelgelemesi, ek kaynaklar, makine elverişliliği sınırlamaları, Lagrange gevşetmesi, tamsayılı programlama, kısıt programlama.

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

1.1 Motivation of the Research

Scheduling is one of the decision-making processes used in manufacturing and service industries. It deals with the allocation of resources to tasks (or jobs) over a given scheduling horizon with the aim of optimizing one or more objectives (Pinedo, 2008, p.1). Scheduling models and algorithms are widely used in manufacturing applications to perform production operations in an efficient way.

A typical scheduling problem in manufacturing systems is defined under three properties: machine configuration, processing characteristics and constraints, and objective function(s).

In terms of machine configurations, four main scheduling environments may be defined: single machine, parallel machines, flow shop, and job shop. Parallel machine scheduling (PMS) is one of the most studied areas in the scheduling literature. It is important from two points of view. From a theoretical point of view, it is a generalization of the single machine, and a special case of the flexible flow shop. From a practical point of view, the occurrence of machines in parallel is common in the real world. Also, PMS techniques are often used in decomposition procedures for multi-stage systems (Pinedo, 2008, p.111). A PMS problem, more formally, can be defined as a system with m machines in parallel and n jobs where each job requires a single operation to be processed on one of the m machines.

PMS problems can further be classified in terms of processing characteristics and constraints. The presence of different processing characteristics and constraints results in different problem classes. In PMS problems, jobs may often not be processed on any of the available machines but rather must be processed on a machine belonging to a specific subset of machines. This situation, called *machine eligibility restrictions*, is widely encountered in real PMS environments.

Furthermore, in most of PMS studies, the only considered resources are machines (Pinedo, 1995; Ventura & Kim, 2003). However, in most real-life manufacturing systems, jobs may also require certain additional resources, such as automated guided vehicles, machine operators, dies, tools, pallets, industrial robots etc (Ventura & Kim, 2003). A common example of additional resources is cross-training workers that perform different tasks associated with different machines (Daniels, Hua & Webster, 1999). Thus, the study of PMS with additional resource constraints is also a significant area of research.

In terms of the objective function, there exist several performance criteria such as mean flow time, completion time of all jobs (i.e., makespan), number of tardy jobs etc.

The research in this dissertation is motivated by a real-world scheduling problem in an injection molding department of an electrical appliance company and deals with a series of resource-constrained parallel machine scheduling problems (RCPMSPs) including machine eligibility restrictions.

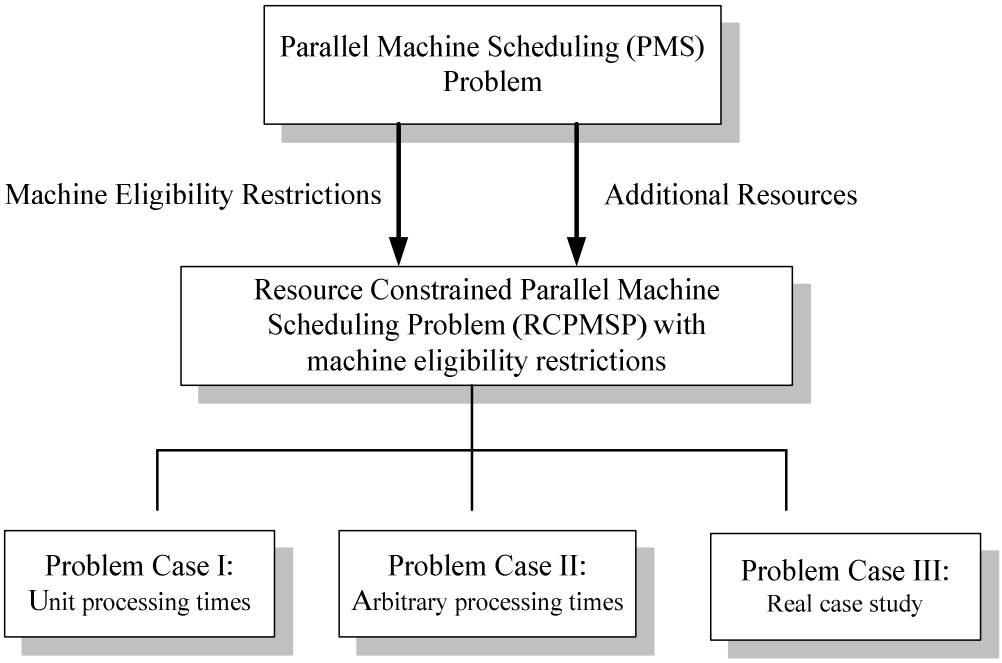


Figure 1.1 Characteristics of the research problems

Figure 1.1 illustrates the characteristics of research problems investigated in this dissertation. The first problem case assumes that processing times of all jobs are equal, while the second and third problem cases allow arbitrary processing times. Different from second problem, the third problem includes the real data with respect to machine eligibility restrictions, number of jobs to be processed, and number of additional resource (i.e., operators) taken from an injection molding department with 36 machines. Details of these research problems will be given in Chapter 4.

1.2 Research Objectives and Methodology

Most of the RCPMSPs are in class of NP-hard problems (see Blazewicz, Lenstra, & Rinnooy Kan, 1983). For most real-world applications, the problem size does not allow to run exact algorithms within a reasonable time limit. Since manufacturers look for rapid, feasible, and easily applicable solutions, this dissertation aims to propose efficient solution algorithms to a series of RCPMSPs with machine eligibility restrictions. Figure 1.2 illustrates the proposed solution approaches with respect to each research problem.

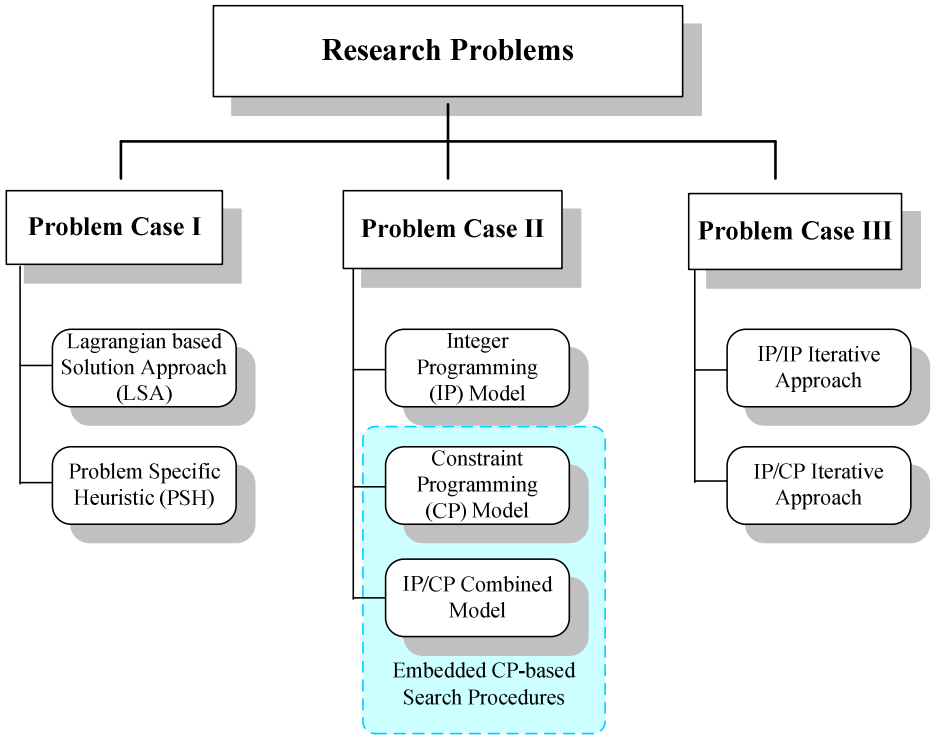


Figure 1.2 Proposed solution approaches for the research problems

For the first problem case, i.e., a RCPMSP with machine eligibility restrictions and unit (equal) processing times with the aim of minimizing total flow time, two solution approaches are proposed. The first one is a Lagrangian-based algorithm which adjusts the infeasible solutions of Lagrangian Relaxation Problem (LRP) to obtain feasible schedules, while the second one is a problem based heuristic. Lagrangian relaxation is also used to obtain tight lower bounds.

For the second problem case, i.e., a RCPMSP with machine eligibility restrictions and arbitrary processing times with the aim of minimizing makespan, three optimization models; an integer programming (IP) model, a constraint-programming (CP) model and a combined IP/CP model are developed. A problem-based search procedure to be used in CP and IP/CP combined models is also proposed to get quick and efficient results.

Finally, for real case problem with its own characteristics and problem size, two solution approaches are proposed. Both approaches are iterative solution methods which partition the problem into loading and scheduling sub-models. In loading phase, an IP model assigns the jobs to machines with the aim of minimizing maximum load on machines. Consequently, scheduling phase uses two alternative models, namely IP and CP, to obtain the final schedule of the jobs subject to additional resource constraints.

The common steps of research methodology followed while dealing with each problem case are given below:

- Characterize the problem structures and investigate efficient model formulations.
- Explore possible solution approaches and generate efficient solution methods for each class of investigated research problems.
- Evaluate the performance of the proposed solution approaches through medium and industrial sized problems with various combinations of problem parameters.
- Compare the results of the proposed solution approaches with existing methods.

1.3 Contributions

In the literature related to RCPMSPs, although a number of studies handle common shared resources, most of them deal with dedicated (i.e., the set of jobs to be processed on each machine is priori known) or identical machines. To the best of our knowledge, no study in this field has considered machine eligibility restrictions.

All the research problems in this dissertation, differently from previous studies, consider machine eligibility restrictions and common shared resource (i.e., machine operators shared by all machines) cases together. This is one of the main contributions of this dissertation.

The other contributions of the dissertation are summarized as follows:

- The studies related to RCPMSPs mainly focus on small sized problems with hypothetical data. Large sized problems, especially the cases encountered in real-life environments, do not receive much attention due to their complex structures.

    - *All research problems in this dissertation are motivated by a real RCPMSP with machine eligibility restrictions encountered in an injection molding department of an electrical appliance company. Moreover, the third problem case also considers real case study with its own large sized real data (i.e., 36 machines, up to 120 jobs and 12 units of additional resource).*

- In the case that a job can only be processed on one of the eligible machines, different flexibility measures of machines become additional parameters of the PMS problem on hand. This situation requires further analysis on different levels of these flexibility measures. So far, the effect of these flexibility measures has only been discussed within classical PMS systems.

    - *This dissertation analyzes the effect of machine eligibility restrictions for the investigated RCPMSPs in terms of two important flexibility measures*

encountered in the PMS literature (Vairaktarakis & Cai, 2003): process flexibility and balance flexibility.

- Since most of RCPMSPs are NP-hard (Blazewicz, Lenstra & Rinnooy Kan, 1983), relaxed formulations of the problems are usually utilized in the literature. This relaxation is generally performed in two ways.

  The first way is to relax some set of constraints in the original formulation. In most mathematical formulations of RCPMSPs, the constraints related to additional resources complicate the problem. By relaxing this set of constraints, the remaining problem probably becomes easy to solve. The common way to utilize such an advantage of relaxation is applying Lagrangian relaxation technique. Although many researchers have studied the use of Lagrangian relaxation algorithms for PMS problems with the aims of both obtaining good lower bounds and producing efficient heuristics based on Lagrangian problem, to the best of our knowledge, only Ventura & Kim (2003) utilizes this technique for a RCPMSP with identical machines and unit processing times. However, they do not consider machine eligibility restrictions.

  - *In this dissertation, a Lagrangian-based solution approach with an efficient heuristic algorithm is proposed for the first problem case. The proposed solution approach not only provides tight lower bounds but also produces efficient results with small optimality gaps.*

  A number of studies (e.g., Grigoriev, Sviredenko & Uetz, 2005, 2006, 2007; Kellerer, 2008) utilize the relaxed (probably solvable) mathematical formulations of the original problem. These relaxed formulations may usually provide individual solutions for a set of sub-problems of the original problem (e.g., resource allocation, job-machine assignment). Then, these individual solutions are adapted to the original problem by applying some greedy heuristic algorithms.

- *For the third problem case, a relaxed (and easily solvable) formulation of the entire problem, i.e., a PMS formulation with machine eligibility restrictions (but without additional resource constraints), is handled to obtain job-machine assignments. Then, with these fixed job-machine assignments, a final schedule with an efficient makespan value may be obtained in a more straightforward way.*

- A common way to present a machine scheduling problem is IP. However, machine scheduling problems are inherently difficult to solve via classical (IP) methods because of their combinatorial nature. When the additional resource constraints are the case, scheduling problems become more complex. For the recent years, constraint programming (CP) has been used as an alternative solution method for solving the combinatorial optimization problems. The studies related to scheduling problems (Darbi-Dowman, Little, Mitra & Zaffalon, 1997; Darbi-Dowman & Little, 1998; Lustig & Puget, 2001; Smith, Brailsford, Hubbard & Williams, 1997) infer that IP seems to be better for problems in which linear programming (LP) relaxations provide strong lower bounds, while CP is better than IP in sequencing, scheduling applications and strict feasibility problems. Since RCPMSPs are natural candidates for strict feasibility problems, CP technique may be utilized individually or as a part of the solution approach for this class of problems. To the best of our knowledge, no study so far utilizes CP technique for solving RCPMSPs.

  - *CP has an advantage in finding quick and efficient results in scheduling problems with resource constraints, especially when these constraints are tight. Therefore, CP technique is utilized in both the second and the third research problems.*

  - *Although CP has an advantage of finding quick and feasible results, it usually lacks proving the optimality when it is used alone. Therefore, for the second problem case, a combined IP/CP model is developed to utilize the complementary strengths of IP and CP techniques. As far as we know, it is*

the first study that uses IP/CP combined model for RCPMSPs. In a related field of PMS with resource constraints, Hooker (2005, 2006) and Chu & Xia (2005) utilize IP and CP models in a decomposition manner and obtain efficient results. However, absence of additional resource(s) other than the main resource (machine) takes us away from classifying these problems into RCPMSPs.

- *For the third problem case, we propose an iterative solution method which partitions the entire problem to loading and scheduling sub-problems to obtain more efficient results. The scheduling sub-problem is solved by IP and also alternatively by CP.*

- *One of the advantages of CP is its ability to use search procedures. By using an efficient search procedure in CP, the search tree can be pruned in the earlier stages, and feasible solutions can be reached in advance. No study so far utilizes problem specific CP-based search algorithms in this class of problems. For the second problem case, two problem specific CP-based search procedures have been proposed to be used in both CP and combined IP/CP models. The efficiency of the proposed search procedures is also confirmed by comparing them with built-in search procedures of OPL (ILOG, 2003) optimization software.*

1.4 Organization of the Dissertation

The rest of this dissertation is organized as follows:

Chapter 2 introduces background issues, notation, and classification of scheduling problems to clarify the scope of our research problems. This chapter also gives a brief review of PMS studies with machine eligibility restrictions and main concepts of resource constrained scheduling related to PMS problems.

Chapter 3 presents a review and discussion of studies related to RCPMSPs by investigating their main characteristics. This chapter also represents the limitations of the existing literature and distinguishing characteristics of the proposed research in this dissertation in terms of both investigated research problems and proposed solution approaches.

Chapter 4 describes the framework of the investigated RCPMSPs, gives main assumptions and defines three problem cases addressed in this dissertation with respect to notation and classification schemes given in Chapter 2. A discussion on the complexity of research problems is provided. This chapter also presents an illustrative example which clarifies the effect of main characteristics, i.e., additional resources and machine eligibility restrictions, of investigated research problems. Finally, a short review of scheduling efforts in the injection molding plants is presented.

Chapter 5 presents an overview of tools employed in the dissertation within two sub-sections. In the first sub-section, Lagrangian relaxation and Lagrangian-based solution approaches are briefly explained. In the second sub-section, IP, CP and IP-CP integration/decomposition schemes are briefly introduced.

The three investigated research problems are studied in detail in Chapter 6, Chapter 7 and Chapter 8, respectively.

Chapter 6 firstly presents an IP model with the objective of minimizing total flow time for the first problem case. Based on this model, a Lagrangian based solution approach with a subgradient optimization procedure has been proposed. Lagrangian relaxation is also used to obtain tight lower bounds. Additionally, a problem specific solution approach is developed to obtain near optimal solutions. Effectiveness of the proposed solution approaches is tested through several test problems with different characteristics.

Chapter 7 deals with the second investigated problem case. Three optimization models; an IP model, a CP model and a combined IP/CP model are developed. Problem-based search procedures to be used in CP and IP/CP combined models are also proposed to get quick and efficient results. The performances of the proposed models are evaluated through randomly generated eight sub-groups of test problems varying in terms of several problem parameters. The efficiency of IP/CP combined model is presented in almost all sub-groups of test problems.

Our third problem case, i.e., real case problem with its own characteristics and problem size, is studied in Chapter 8. In order to obtain efficient results, an iterative solution method which partitions the problem into loading and scheduling sub-models is proposed. In loading phase, an IP model is used to assign the jobs to machines. In scheduling phase, two alternative models, namely IP and CP, are used to obtain the final schedule of the jobs. Consequently, the proposed solution approaches are applied to a set of problems with real data and their performances are evaluated.

Finally, Chapter 9 gives the concluding remarks, represents the contributions and identifies future directions of the proposed research.

CHAPTER TWO
BACKGROUND

2.1 Introduction

The scheduling function in a production system interacts with many other functions. The diagram in Figure 2.1 depicts the information flow in a manufacturing system. Notice that capacity status and scheduling constraints are determined by the decisions made at the top of the hierarchy. Thus, scheduling performance is directly restricted by these decisions.

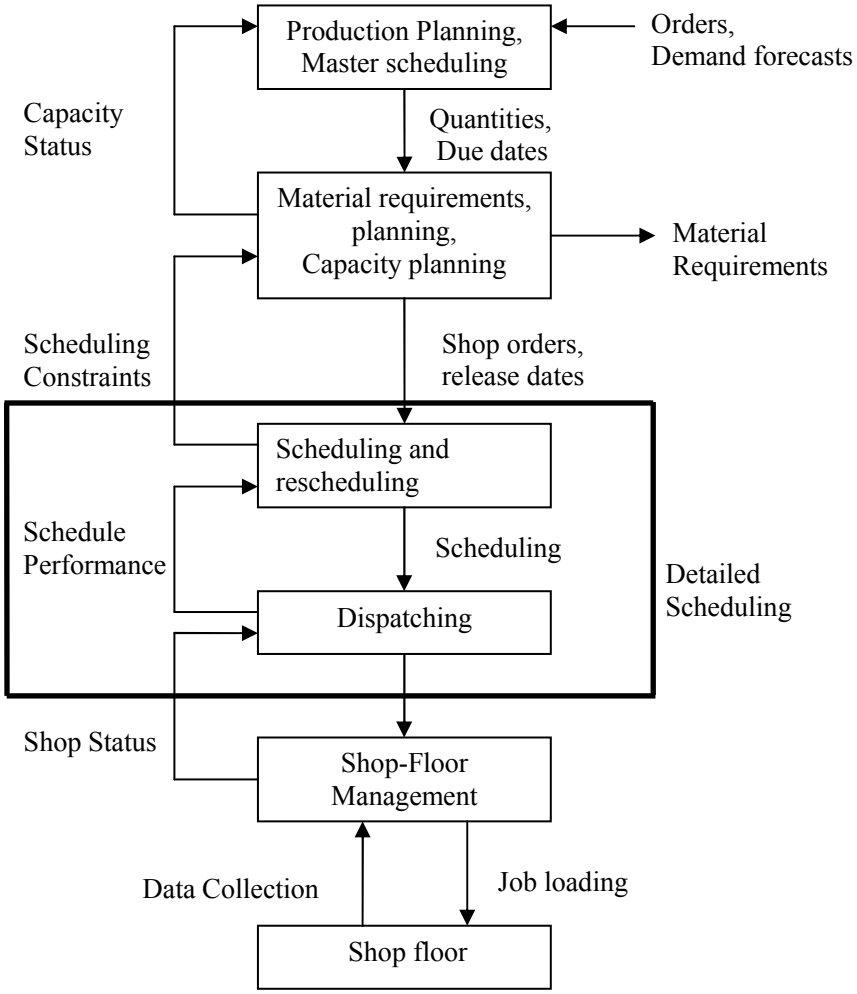


Figure 2.1 Information flow diagram in a manufacturing system (Pinedo, 1995, p.4)

As seen from Figure 2.1, the decisions that are made in the production planning process and shop floor control may have an impact on scheduling. At the production planning process, the inventory levels, forecasts, due dates, capacity constraints and resource requirements have to be considered. On the other hand, some unexpected events on the shop floor such as machine breakdowns or processing times that are longer than anticipated have to be also taken into account while building schedules. Therefore, the schedule is built based on all these restrictions.

Scheduling involves allocation of machinery and other resources (e.g., labor, tooling) to different orders, each of which is referred to as a job. A proper allocation of resources enables the company to optimize its objectives and achieve its goals. The objectives may take many forms, such as minimizing the time to complete all tasks or minimizing the number of tasks completed after their due dates (Pinedo & Chao, 1999, p.2).

Defining a machine scheduling problem in words is often easy: A system with n jobs that will be processed on one, or a subset or all of m machines probably further requiring additional resources (e.g., machine operators, tools, pallets) and should be completed subject to other system constraints to meet some objectives.

Unfortunately, scheduling efforts are often difficult to perform and implement. Since the time function goes into the scheme, solution branches grow up to a huge amount, at once. Then, implementation difficulties arise related to the modelling of the real-world scheduling problems whereas technical difficulties come across the solution methodology and procedures. Resolving these difficulties takes skill and experience but is often financially and operationally well worth the effort (Pinedo & Chao, 1999, p.5).

This dissertation deals with a series of PMS problems subject to additional resource constraints and machine eligibility restrictions. This chapter, therefore, gives background information related to the research problems in this dissertation.

The subsequent sections of this chapter are organized as follows. Section 2.2 gives the notation to be used throughout the dissertation. Section 2.3 presents the classification of scheduling problems giving further attention to the issues related to our research problems. Section 2.4 clarifies the concept of machine eligibility restrictions and reviews the related studies within the framework of PMS. A brief introduction to additional resources is presented in Section 2.5. Finally, Section 2.6 summarizes this chapter presenting its relation to the subsequent chapters of the dissertation.

2.2 Notation

The following notation will be used throughout the dissertation. Additional notation will be defined when required:

n	number of jobs
N	set of jobs
m	number of machines
M	set of machines
i	index of jobs, $i = 1, \ldots, n$
j	index of machines, $j = 1, \ldots, m$
T	number of time periods in the scheduling horizon
t	index of time periods, $t = 1, \ldots, T$
p_{ij}	processing time of job i on machine j.
p_i	processing time of job i (independent of machine j)
r_i	the earliest time at which job i can start its processing, also known as ready (release) time.
d_i	due date of job i
w_i	weight of job i, denoting the importance of job i relative to other jobs.
C_i	completion time of job i
$C_{\max}$	maximum completion time of all jobs in the system, i.e., makespan.

v_j speed of machine j (defined for uniform machines)

v least common multiple of $v_1, v_2, \ldots, v_m$

$v_{\max}$ $\max\{ v_1, v_2, \ldots, v_m \}$

v_{ij} speed of job i on machine j (defined for unrelated machines)

2.3 Classification of Scheduling Problems

We will use a classification scheme introduced by Graham, Lawler, Lenstra & Rinnooy Kan (1979) and Blazewicz, Lenstra & Rinnooy Kan (1983). The scheme employs a three-field classification $\alpha|\beta|\gamma$ where

- the first field α specifies the machine environment
- the second field β represent job characteristics, and
- the third field γ denotes the objective function.

The following subsections present information on the components of this classification scheme. Note that, the list of related components is not comprehensive, but mainly includes the ones related to the investigated research problems. For a comprehensive list see Graham et al. (1979) and Blazewicz, Lenstra & Rinnooy Kan (1983).

2.3.1 Machine Environment

The first field $\alpha = \alpha_1\alpha_2$ specifies the machine environment.

$\alpha_1 \in \{\varnothing, P, Q, R, PD\}$ illustrates the type of machine arrangement:

$\alpha_1 = \varnothing$: single machine,

$\alpha_1 = P$: identical parallel machines,

$\alpha_1 = PD$: parallel dedicated machines,

$\alpha_1 = Q$: uniform parallel machines,

$\alpha_1 = R$: unrelated parallel machines.

$\alpha_2 \in \{\varnothing, k\}$ denotes the number of machines in the problem:

$\alpha_2 = \varnothing$: the number of machines assumed to be variable (i.e., a part of input)

$\alpha_2 = k$: the number of machines is equal to *k*.

In terms of α_1 field, four parallel machine sub-cases are distinguished:

Identical parallel machines (*P*)**.** Job *i* requires a single operation and may be processed on any one of the *m* machines with the same processing time p_i.

Uniform parallel machines (*Q*)**.** Machines have different speeds; the speed of machine *j* is denoted by v_j and determined as proportional (relative) to speeds of other machines. The processing time for a job assigned to a machine, is equal to the job processing time divided by machine speed, $p_{ij} = p_i / v_j$. If all machines have the same speed, then the environment is reduced to identical parallel machines.

Unrelated machines (*R*)**.** There are *m* different machines in parallel, and there is no particular relationship among processing times for each job. Machine *j* can process job *i* at speed v_{ij}. The time p_{ij} is equal to p_i / v_{ij}.

Parallel dedicated machines (*PD*)**.** The set of jobs that will be processed on each machine is pre-determined. More formally, N_j represents the set of jobs that must be processed on machine *j*. Note that, the jobs are partitioned into *m* disjoint subsets; i.e., $N_j \cap N_{j'} = \varnothing$, for all $j \neq j'$ and $\bigcup_{j \in M} N_j = N$. This situation eliminates the job-machine assignment sub-problem.

On the other hand, if job *i* is not allowed to be processed on just any machine, but is allowed to be processed on a given subset of machines, say subset M_i, then the entry M_i appears in the β field. This M_i entry defines machine eligibility restrictions.

2.3.2 Processing Characteristics and Constraints

The field $\beta \subset \{\beta_1,..., \beta_7\}$ specifies characteristics of the jobs or of the resources.

$\beta_1 \in \{\varnothing, pmtn\}$ indicates whether there exists the possibility of preemption.

$\beta_1 = \varnothing$: no preemption is allowed.

$\beta_1 = pmtn$: preemption is allowed.

$\beta_2 \in \{\varnothing, res\lambda\sigma\delta\}$ characterizes additional resources (Blazewicz, Lenstra & Rinnooy Kan, 1983):

$\beta_2 = \varnothing$: no additional resource constraints

$\beta_2 = res\lambda\sigma\delta$: there are specified resource constraints, where λ, σ and δ are characterized as follows:

- If λ is a positive integer, then the *number of resource types* is constant and equal to λ; if $\lambda = \bullet$, then it is part of the input and arbitrary.
- If σ is a positive integer, then all *resource sizes* are constant and equal to σ; if $\sigma = \bullet$, then all resource sizes are arbitrary.
- If δ is a positive integer, then all *resource requirements* have a constant upper bound equal to δ; if $\delta = \bullet$, then no such bounds are specified.

$\beta_3 \in \{\varnothing, prec, tree, chain\}$ reflects the precedence constraints:

$\beta_3 = \varnothing$: independent jobs,

$\beta_3 = prec$: general precedence constraints,

$\beta_3 = tree$: precedence constraints forming a tree

$\beta_3 = chain$: precedence constraints forming a chain.

$\beta_4 \in \{\varnothing, r_i\}$ describes ready times:

$\beta_4 = \varnothing$: all ready times are equal to zero,

$\beta_4 = r_i$: ready times differ per job *i*.

$\beta_5 \in \{p_i = p, p_i\}$ describes job processing times:

$\beta_5 = (p_i = p)$: all jobs have processing times equal to *p* units.

$\beta_5 = p_i$: jobs have arbitrary processing times. (p_i is not necessarily used in the representation)

$\beta_6 \in \{\varnothing, d_i = d, d_i\}$ describes job due dates:

$\beta_6 = \varnothing$: jobs have no due dates.

$\beta_6 = (d_i = d)$: all jobs have a common due date *d*.

$\beta_6 = d_i$: jobs have distinct due dates.

$\beta_7 \in \{\varnothing, M_i\}$ refers to machine eligibility restrictions:

$\beta_7 = \varnothing$: all machines are eligible for all jobs.

$\beta_7 = M_i$: job *i* can only be processed on a specific machine subset, M_i.

Note that the symbol '$\varnothing$' is not necessarily used in the representation of corresponding β fields.

2.3.3 Objective Function

The third field γ refers to objective function. The optimality criteria are built by considering the following elementary functions:

flow time: $F_i = C_i - r_i$;

lateness: $L_i = C_i - d_i$;

tardiness: $T_i = \max\{C_i - d_i, 0\}$;

earliness: $E_i = \max\{d_i - C_i, 0\}$;

unit penalty: $U_i = \begin{cases} 1 & \text{if } C_i > d_i \\ 0 & \text{otherwise.} \end{cases}$

The most significant minsum objective functions are:

$\sum_{i=1}^{n} C_i$: total completion time; $\sum_{i=1}^{n} w_i C_i$: total weighted completion time;

$\sum_{i=1}^{n} F_i$: total flow time; $\sum_{i=1}^{n} w_i F_i$: total weighted flow time;

$\sum_{i=1}^{n} T_i$: total tardiness; $\sum_{i=1}^{n} w_i T_i$: total weighted tardiness;

$\sum_{i=1}^{n} U_i$: number of tardy jobs; $\sum_{i=1}^{n} w_i U_i$: weighted number of tardy jobs.

The most significant minmax objective functions are:

$L_{\max} = \max_i L_i$: maximum lateness;

$T_{\max} = \max_i T_i$: maximum tardiness;

$C_{\max} = \max_i C_i$: makespan.

2.3.4 Complexity Hierarchy

A solution algorithm for a scheduling problem can be often applied to another scheduling problem as well. For instance, $1 \mid\mid \sum C_i$ is a special case of $1 \mid\mid \sum w_i C_i$ and an algorithm for $1 \mid\mid \sum w_i C_i$, can also be used for $1 \mid\mid \sum C_i$. In complexity terminology it is then said that $1 \mid\mid \sum C_i$ *reduces* to $1 \mid\mid \sum w_i C_i$ and this is denoted as (Pinedo, 2008, p.26):

$$1 \mid\mid \sum C_i \propto 1 \mid\mid \sum w_i C_i$$

Pinedo (2008) presents the complexity hierarchy of deterministic scheduling problems in three field classification scheme. Figure 2.2 illustrates an adapted

form of these complexity hierarchy relations given by Pinedo (2008, p.26). Note that, we have placed *m* parallel dedicated machines, i.e., "*PDm*", between single machine, "1", and *m* parallel identical machines, "*Pm*".

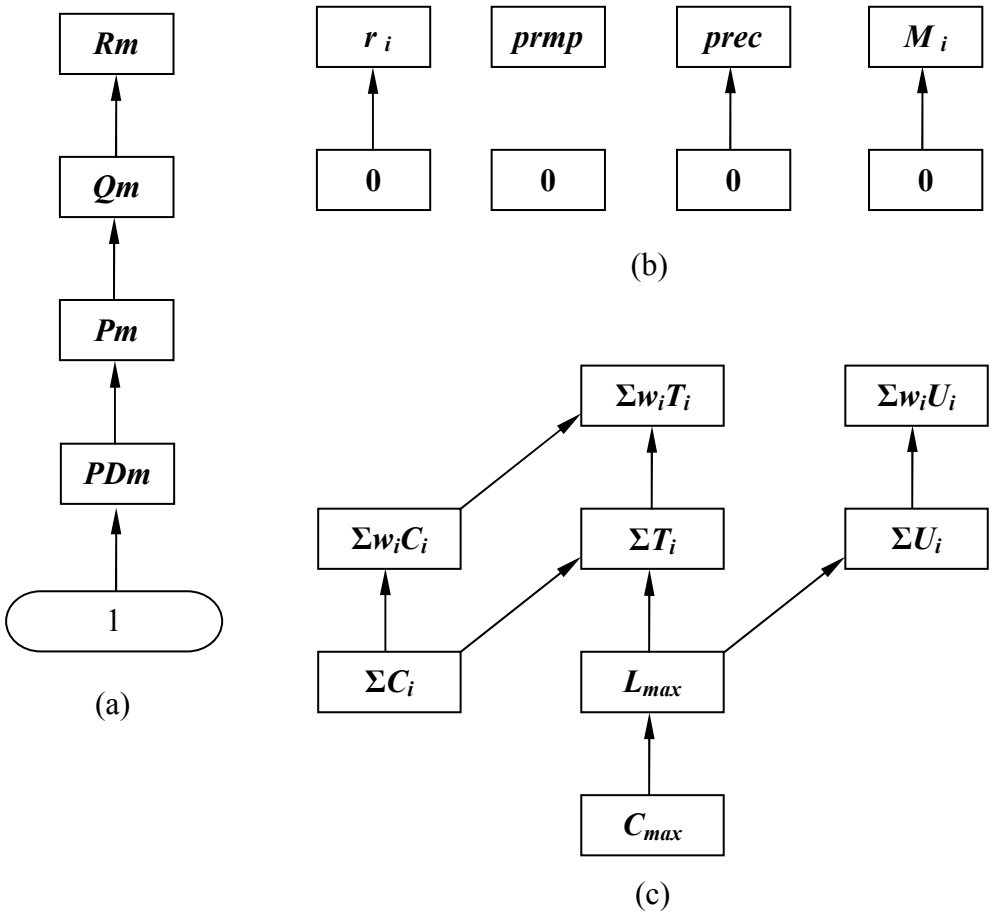


Figure 2.2 Complexity hierarchies of deterministic scheduling problems: (a) Machine environments (b) Processing restrictions and constraints (c) Objective functions (Adapted from Pinedo, 2008, p.26)

Based on the complexity hierarchy, a chain of reductions can be given as follows:

$$1 \mid\mid \sum C_i \;\propto\; Pm \mid\mid \sum C_i \;\propto\; Pm \mid M_i \mid \sum C_i$$

Once the notation and background issues have been given, the following two sections introduce the concepts of machine eligibility restrictions and additional resources within the PMS framework.

2.4 Parallel Machine Scheduling with Machine Eligibility Restrictions

Identical PMS is a system of m parallel identical machines each of which is capable of executing any job. However, in a parallel machine environment, job i may not be processed on just any one of m machines in parallel, but rather has to be processed on a machine that belongs to a specific subset M_i of the machines (Pinedo & Chao, 1999, p.19). This may occur, as described earlier, when the machines in parallel are not exactly identical. This situation, named "machine eligibility restrictions" is widely encountered in real scheduling environments.

More formally, we are given a set of n jobs and a set of m parallel machines, each job i has a processing time p_i and a specific subset of machines, M_i, to which it can be assigned to optimize some objective function (say $C_{\max}$). In terms of the three-field notation of Graham et al. (1979), this problem is denoted as P $|M_i|$ $C_{\max}$ if the machines are identical, Q $|M_i|$ $C_{\max}$ if the machines are uniform, and R $|M_i|$ $C_{\max}$ if the machines are unrelated. R$|M_i|$ $C_{\max}$ can be viewed as a special case of R $||$ $C_{\max}$ since we can set processing time of job i on machine j to infinity if $j \notin M_i$. Therefore, R $|M_i|$ $C_{\max}$ is equivalent to R $||$ $C_{\max}$ (Leung & Li, 2008).

Scheduling with machine eligibility restrictions have extensively studied under different names (Leung & Li, 2008). These are "multi-purpose machine scheduling" (e.g., Brucker, 2004; Vairaktarakis & Cai, 2003), "scheduling with processing set restrictions" (e.g., Leung & Li, 2008; Glass & Kellerer, 2007) and "scheduling with machine eligibility restrictions" (e.g., Centeno & Armacost, 1997; Centeno & Armacost, 2004; Edis, Araz & Ozkarahan, 2008). The term of "machine eligibility restrictions" is going to be used throughout the dissertation. An extensive survey on PMS problems with machine eligibility restrictions can be found in Leung & Li (2008). On the other hand, machine eligibility restrictions are also encountered in multiprocessor task scheduling problems (i.e., P$|$ set_j $|C_{\max}$) where a task requires more than one processor at a time. The entry "set_j" states that each task can be processed on exactly one subgraph of the multiprocessor system (see Blazewicz, Ecker, Pesch, Schmidt & Weglarz, 2007a).

In terms of machine eligibility restrictions, there are two important special cases that have considerable attention in the literature: *nested* and *inclusive* sets.

The M_i sets are nested when one and only one of the following four conditions holds for each pair of jobs *i*, *k* (Pinedo, 1995, p.70):

1. M_i is equal to M_k $(M_i = M_k)$
2. M_i is a subset of M_k $(M_i \subset M_k)$
3. M_k is a subset of M_i $(M_k \subset M_i)$
4. M_i and M_k do not overlap $(M_i \cap M_k = \varnothing)$

Inclusive set, on the other hand, is a special case of nested set in that only first three conditions above are in order for each pair of jobs *i*, *k*. Leung & Li (2008) illustrate these two special cases (in case of identical machines) as $P|M_i(\text{nested})|C_{\max}$ and $P|M_i(\text{inclusive})|C_{\max}$ in the three-field notation.

Pinedo (1995, p.71) proved that the least flexible job first (LFJ) rule is optimal for $Pm \mid p_i = 1, M_i(\text{nested}) \mid C_{\max}$. Every time a machine is idle, LFJ rule chooses the job that can be processed on the smallest number of machines. Since M_i sets have to be nested for two-machine problem with unit processing times, i.e., $P2 \mid p_i = 1, M_i \mid C_{\max}$, LFJ rule always gives the optimal solution (Pinedo, 1995, p.71). Pinedo (1995, p.81) proved that LFJ rule is also optimal for $Pm \mid p_i = 1, M_i(\text{nested}) \mid \sum C_i$.

On the other hand, LFJ rule does not specify which machine should be considered first when a number of machines are free simultaneously. For such cases, Pinedo (1995, p.71) states that "It is advantageous to consider first the least flexible machine. The flexibility of a machine could be defined as the number of remaining jobs that can be processed (or the total amount of processing that can be done) on the machine." The least flexible machine (LFM) rule can be used to select the machine that can process the smallest number of jobs and assign that machine to the least flexible job that can be processed on it. Any ties can be broken arbitrarily. Pinedo (1995) referred to this heuristic as LFM-LFJ.

For more general cases with unit processing times where M_i sets are neither inclusive nor nested, a number of researchers have developed polynomial time exact algorithms. Table 2.1 presents these problems and complexity of the proposed solution algorithms. Lin & Li (2004) and Harvey, Ladner, Lovasz & Tamir (2006) study the problem $P \mid p_i = 1, M_i \mid C_{\max}$ and independently develop polynomial time algorithms. Lin & Li (2004)'s algorithm can also be applicable to $Q \mid p_i = 1, M_i \mid C_{\max}$. Li (2006) study the variants of this problem with various objective functions and uniform machines.

Table 2.1 Problems solvable in polynomial time

Problem	Algorithm Complexity	Reference
$P \mid p_i = 1, M_i \mid C_{\max}$	$O(n^3 \log n)$	Lin & Li (2004)
$Q \mid p_i = 1, M_i \mid C_{\max}$	$O(n^3 \log nv)$	Lin & Li (2004)
$P \mid p_i = 1, M_i \mid C_{\max}$	$O(n^2 m)$	Harvey et al. (2006)
$Pm \mid p_i = 1, M_i \mid C_{\max}$	$O(n^2 (m + \log n) \log n)$	Li (2006)
$Pm \mid p_i = 1, M_i \mid \sum C_i$	$O(n^{2.5} m \log n)$	Li (2006)
$Qm \mid p_i = 1, M_i \mid C_{\max}$	$O(n^2 (m + \log nv_{\max}) \log n)$	Li (2006)
$Qm \mid p_i = 1, d_i, M_i \mid \sum U_i$	$O(n^{2.5} m \log n)$	Li (2006)
$Qm \mid p_i = 1, d_i, M_i \mid \sum f_i(T_i)$	$O(n^3 m)$	Li (2006)
$Qm \mid p_i = 1, d_i, M_i \mid \max\{f_i(T_i)\}$	$O(n^{2.5} m \log n)$	Li (2006)

In real cases, however, the processing times are often arbitrary. Since $P \parallel C_{\max}$ is NP-hard (Garey & Johnson, 1979), $P|M_i(\text{inclusive})|C_{\max}$, $P|M_i(\text{nested})|C_{\max}$, and $P|M_i|C_{\max}$ are NP-hard as well Leung & Li (2008). Because of the NP-hardness of the problem, the studies in the literature focus on approximation algorithms. To evaluate the performance of various approximation algorithms, worst case ratio is used as a common criterion. An algorithm with a worst case performance ratio of ρ is described as $\rho - approximation$ algorithm. For instance, if an algorithm has a worst case performance ratio of two, it is named as $2 - approximation$ algorithm, and a solution value is guaranteed to be no more than twice the optimum, regardless to the input data. Table 2.2 summarizes the solution approaches related to this class of problems. Glass & Kellerer (2007) propose polynomial time approximation algorithms for both nested and inclusive sets. For inclusive sets, Ou, Leung & Li (2008) propose a better polynomial time algorithm which

improves the worst case ratio presented by Glass & Kellerer (2007). Moreover, Ou, Leung & Li (2008) develop a polynomial time approximation scheme (PTAS) for the same problem. Note that, *PTAS* is a family of algorithms that has polynomial time running in the length of the problem input and delivers a worst-case ratio bound of $1+\varepsilon$, where $\varepsilon>0$ and can be set arbitrarily close to zero. Later, Li & Wang (2009) deal with an extended version of this problem incorporating release times, i.e., $P|M_i$(inclusive), $r_i|C_{max}$, and develop a PTAS for it. Ji & Cheng (2008) also present a fully polynomial-time approximation scheme (FPTAS) for the special case $Pm|M_i$(inclusive)$|C_{max}$. Note that a *PTAS* is called FPTAS, if the running time is polynomial in also $1/\varepsilon$.

Table 2.2 Polynomial Time Approximation Algorithms

Problem	**Solution Algorithm**	**Worst Case Ratio**	**Reference**
$P\|M_i$(nested)$\| C_{max}$	Strongly Polynomial Time	2-1/m	Glass & Kellerer (2007)
$P\|M_i$(inclusive)$\| C_{max}$	Polynomial Time	3/2	Glass & Kellerer (2007)
$P\|M_i$(inclusive)$\| C_{max}$	Polynomial Time	4/3	Ou, Leung & Li (2008)
$P\|M_i$(inclusive)$\| C_{max}$	PTAS	-	Ou, Leung & Li (2008)
$P\|M_i$(inclusive), $r_i\| C_{max}$	PTAS	-	Li & Wang (2009)
$Pm\|M_i$(inclusive)$\| C_{max}$	FPTAS	-	Ji & Cheng (2008)
$R\|\|C_{max}$ ($\approx P\|M_i\| C_{max}$)	Polynomial Time	2	Lenstra, Shmoys & Tardos (1990)
$R\|\|C_{max}$ ($\approx P\|M_i\| C_{max}$)	Polynomial Time	2-1/m	Shchepin and Vakhania (2005)

Recall that, in practical cases, the processing times are often arbitrary, and M_i sets are not nested. Lenstra, Shmoys & Tardos (1990) and Shchepin & Vakhania (2005) proposed polynomial time approximation algorithms for R || C_{max} which is also equivalent to P| M_i | C_{max}. Other than these studies, Vairaktarakis & Cai (2003) propose a branch-and-bound (B&B) algorithm to solve $P|M_i| C_{max}$ problem optimally. They stated that B&B algorithm is able to solve problem instances for up to 50 jobs. The authors also developed a number of heuristic algorithms as well as lower bounds and compared the performances of heuristics empirically. They also assess the value of flexibility as compared with fully flexible parallel machines and found that very small amounts of flexibility appropriately

distributed across machines provide nearly the same makespan performance as system of fully flexible parallel machines.

Centeno & Armacost (1997) developed a heuristic algorithm based on Pinedo's LFJ-LFM rule for the problem $Pm|M_i, r_i|L_{\max}$ for the special case where due dates are equal to release dates plus a constant, making the $L_{\max}$ equivalent to $C_{\max}$. In another study, Centeno & Armacost (2004) show that longest processing time first (LPT) rule performs better than LFJ or LFM-LFJ rule when M_i sets are not nested and arbitrary processing times are considered.

Another significant point that requires further attention is the flexibility measures of machines. Surely, the presence of machine eligibility restrictions makes the production environment less flexible in comparison to identical PMS environment where all machines are capable of processing every job. In case of machine eligibility restrictions, machines have a level of flexibility determined by *number* and *distribution* of ones in the availability matrix $\boldsymbol{A}$ where A_{ij} is equal to one if machine j is eligible to process job i, (i.e., $j \in M_i$); and zero otherwise. Vairaktarakis & Cai (2003) define *process flexibility index* by capturing the number of ones in $\boldsymbol{A}$:

$$F_{\mathrm{P}} = \frac{\sum_{i,j} A_{ij} - n}{n(m-1)}$$

Note that, each job should be able to be processed on at least one machine to guarantee feasible schedules. Vairaktarakis & Cai (2003) also state that there are many different configurations of matrix $\boldsymbol{A}$**,** for a given value of F_{P}, depending on the distribution of flexible characteristics onto the machines. In order to capture these configurations, they introduce a measure of *flexibility balance*, F_{B}:

$$F_{\mathrm{B}} = \frac{1}{m}\sum_{j=1}^{m}\left|\sum_{i=1}^{n} A_{ij} - \overline{B}\right|$$

where $\overline{B} = \frac{\sum_{i,j} A_{ij}}{m}$ is the average number of jobs per machine. Since different

jobs require distinct machine flexibilities, $\overline{B}$ also reflects the level of flexibility of an average machine.

Let us clarify these two measures by representing an example given in Vairaktarakis & Cai (2003). Suppose that a PMS problem with n = 8 jobs, m = 4 machines and the following two different availability matrices is given:

$$A_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}; \; A_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Then, for $\boldsymbol{A_1}$: $F_{\mathrm{P}} = \dfrac{16-8}{8(4-1)} = 0.33$; $\overline{B} = 4$; $F_{\mathrm{B}} = \dfrac{1}{4}(|-3|+|-3|+|3|+|3|) = 3$.

For $\boldsymbol{A_2}$: $F_{\mathrm{P}} = \dfrac{16-8}{8(4-1)} = 0.33$; $\overline{B} = 4$; and $F_{\mathrm{B}} = \dfrac{1}{4}(0+0+0+0) = 0$.

Note that, $\boldsymbol{A_1}$ corresponds to a system where processing flexibility is concentrated on only third and fourth machines; while the first and second machines can process only one job. On the other hand, F_{B} value associated with $\boldsymbol{A_2}$ reflects the most balanced distribution of flexibility among all configurations with same F_{P}. Surely, there are still several other configurations with the same F_{P} and F_{B} values.

Consequently, the pair (F_{P}, F_{B}) describes flexibility in PMS problems with machine eligibility restrictions in significant detail (Vairaktarakis & Cai, 2003).

While generating the test instances related to our research problems, these flexibility measures are going to be used in order to incorporate and analyze the effect of machine eligibility restrictions.

The next section briefly identifies resource constraints in the scheduling problems, which exploits the key characteristic of our research problems.

2.5 Parallel Machine Scheduling with Additional Resources

This section introduces the main characteristics of resource constrained machine scheduling problem. A detailed literature review on the studies related to the RCPMSPs is presented in Chapter 3.

The idea of RCPMSPs goes back to early 1970s. It has been studied in case of scheduling the tasks on parallel processors on a time-sharing computer system where mass storage, primary memory and data channels can be treated as additional resources (Gonzalez, 1977; Krause, Shen & Schwetman, 1975). To specialize the issue for PMS problems, consider that, a set of n jobs, a set of m identical parallel machines, and a set of resource types $\boldsymbol{R}=\{R_1, R_2, \ldots, R_u\}$, which are available in the amounts of b_1, b_2,…, b_u units, respectively, are given. The string $\boldsymbol{R(i)}$ represents the amount of additional resources required by job i, and can be expressed as follows:

$\boldsymbol{R(i)}=[R_1(i), R_2(i), \ldots, R_u(i)]$ where $R_k(i) \le b_k$, $k = 1, \ldots, u$ denotes the number of units of resource R_k required by job i.

The general resource-constrained scheduling problem involves scheduling a set of jobs over a discrete time horizon, where each job requires some constant amount of a limited resource over its processing time. Resource-constrained scheduling problems are difficult, due to the fact that besides the efficient allocation of jobs, it is also necessary to consider feasible grouping of simultaneously processed tasks that will use resources within their availability limits at each point in time. Figure 2.2 illustrates the effect of resource constraints in a simple scheduling problem with n = 2 jobs, m = 2 machines (i.e., M1 and M2) and one additional resource type $\boldsymbol{R}=\{R_1\}$ with unit size (b_1 = 1) and one resource requirement for each job, i.e., $R_1(1) = 1$, $R_1(2) = 1$. Figure 2.2 (a)

demonstrates an infeasible schedule where the constraint on the available number of resources is violated for some interval of time periods due to the overlapping of two jobs. If beginning time of job on M2 is delayed to the completion time of job on M1, a feasible schedule is obtained (see Figure 2.2(b)).

Blazewicz, Brauner & Finke (2004) state that if the resources are needed together with a processor (machine) during the processing of a given task set, then the resources is called *processing resources*. Otherwise, i.e., if the resource is needed either before the processing of a task or after it, then the resources is called *input-output resources*.

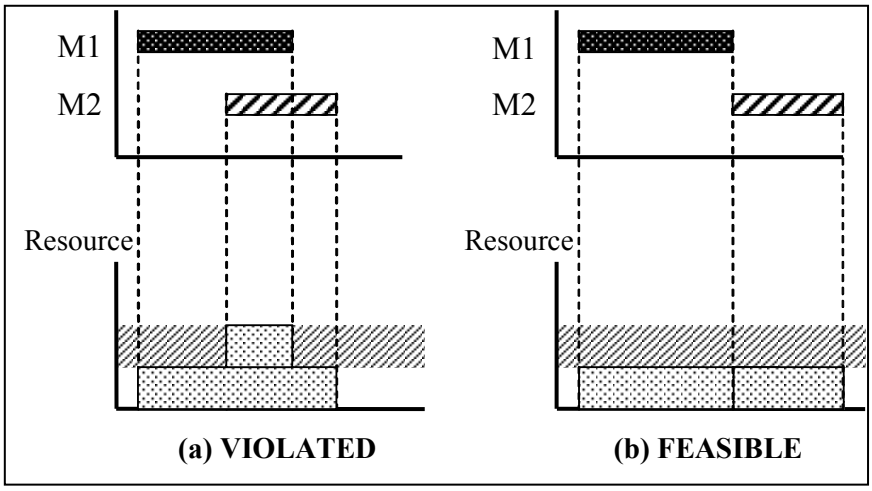


Figure 2.2 Effect of additional resource constraints on scheduling

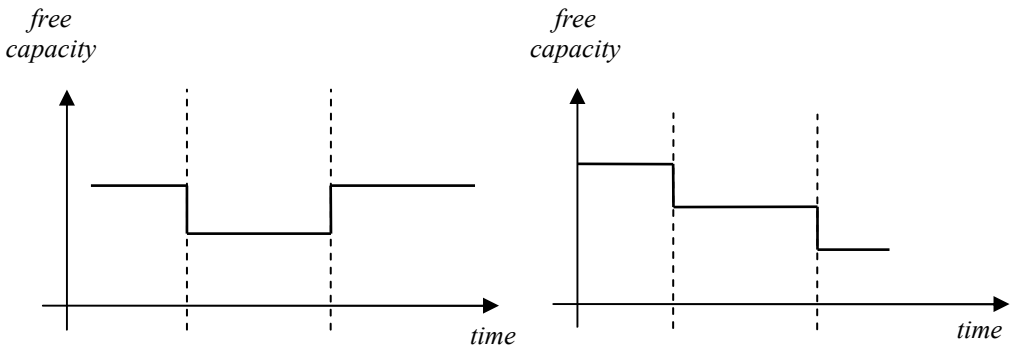
The additional resources are also classified in two points of view: *resource constraints* and *resource divisibility* (Blazewicz et al., 2007b; Slowinski, 1980).

From the viewpoint of resource constraints:

- a resource is *renewable*, if only its total usage at every moment is constrained. In other words, once it is used for a task, it may be used again for another task after being released from this task.
- a resource is *nonrenewable*, if its total consumption is constrained. In other words, once it is used by some task, it cannot be assigned to any other task.

- a resource is *doubly constrained*, if it is both renewable and non-renewable. In other words, both total usage and total consumption are constrained.

Figure 2.3 illustrates the changes in the level of renewable and non-renewable resources through the time horizon, respectively.



(a) renewable resource (b) non-renewable resource

Figure 2.3 Renewable *vs*. Non-Renewable resources

In scheduling problems, the additional resources are usually renewable (e.g., machines, tools, pallets, and operators); while non-renewable resources such as budget, raw materials, energy, frequently occur in planning problems.

From the viewpoint of resource divisibility:

- *Discrete resources* can be allocated to tasks in discrete amounts from a given finite set of possible allocations.
- *Continuous resources* can be allocated to tasks in arbitrary amounts from a given interval.

Blazewicz, Lenstra & Rinnooy Kan (1983) and Blazewicz et al.(2007b) outlined a range of initial results on the complexity of resource constrained scheduling problems and classified the resource constraints in six different types, some of which are obvious generalization of others. Figure 2.4 illustrates these six types and simple transformations between them in terms of $res\lambda\sigma\delta$ classification given in Section 2.3.2. An arc from type (a) to type (b) indicates that (a) is a

special case of (b). Obviously, the most general version of resource constraints is "$res\cdots$".

All but one of these transformations are quite obvious (Blazewicz et al., 2007b). The transformation $(res1\cdot\cdot)\rightarrow(res\cdot 11)$ has been proved in Blazewicz, Cellary, Slowinski, & Weglarz (1986).

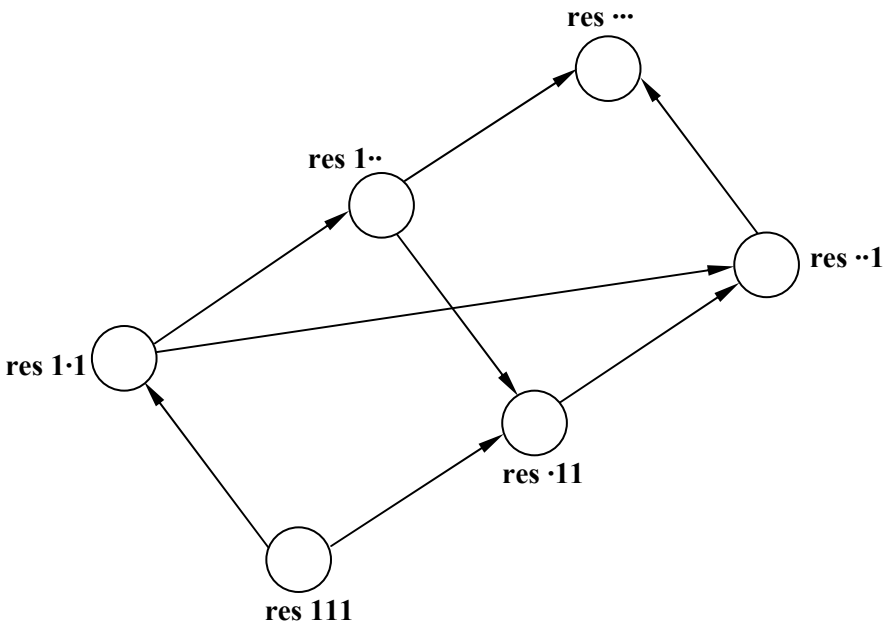


Figure 2.4 Reductions between types of resource constraints (Blazewicz et al., 2007b)

2.6 Chapter Summary

This chapter summarized background information on the research problems studied in this dissertation. Firstly, the role of scheduling in a general manufacturing system has been introduced. Then the notation used throughout this chapter has been presented. The common classification scheme $\alpha|\beta|\gamma$ has also been represented with the extensions of properties related to our research problems. Finally, two main characteristics of our research problems, i.e., machine eligibility restrictions and additional resource constraints, have been explained briefly.

Once the background information regarding the research problems has been provided by this chapter, Chapter 3 presents a comprehensive review of PMS problems with additional renewable resources and Chapter 4 introduces three investigated research problems with their own characteristics.

CHAPTER THREE
PARALLEL MACHINE SCHEDULING WITH ADDITIONAL RESOURCES: LITERATURE REVIEW AND DISCUSSION

3.1 Introduction

Scheduling models and algorithms are most widely used in manufacturing applications to perform production in an efficient way. PMS problems are one of the most studied areas in the scheduling literature. Cheng & Sin (1990) give a comprehensive review on PMS research. Mokotoff (2001) presents an overview of the research on the case of optimal makespan on identical parallel machines. In a more recent paper, Pfund, Fowler & Gupta (2004) survey the literature related to traditional unrelated parallel machine deterministic scheduling problems.

In most of PMS studies, the only considered resource is the machine. However, in most real-life manufacturing environments, jobs may also require, besides machines, certain *additional resources*, such as automated guided vehicles, machine operators, tools, pallets, dies, industrial robots etc., for their handling and processing. (Blazewicz, Lenstra & Rinooy Kan, 1983; Slowinski, 1980; Ventura & Kim, 2000). Thus, the study of PMS with additional resource constraints is a significant area of research. This chapter gives a review and discussion of studies related with RCPMSPs by investigating their main characteristics. The strengths and weaknesses of the literature, open areas and future needs of the related studies are also given. An earlier version of this chapter can be found in Edis & Ozkarahan (2007).

Figure 3.1 illustrates the classification of additional resources as detailed in Section 2.5. In this chapter, we only deal with *processing resources* that are *discrete* and *renewable*. For the studies relating input/output resources, interested readers are directed to Blazewicz, Brauner & Finke (2004), Hall, Potts & Sriskandarajah (2000) and Glass, Shafransky & Strusevich (2000). Comprehensive studies on continuous resources can be found in Jozefowska & Weglarz (2004) and Blazewicz et al. (2007b). A study related to non-renewable resources is given by Shabtay & Kaspi

(2006). Finally, Ozdamar & Ulusoy (1994) deal with a doubly constrained project scheduling problem.

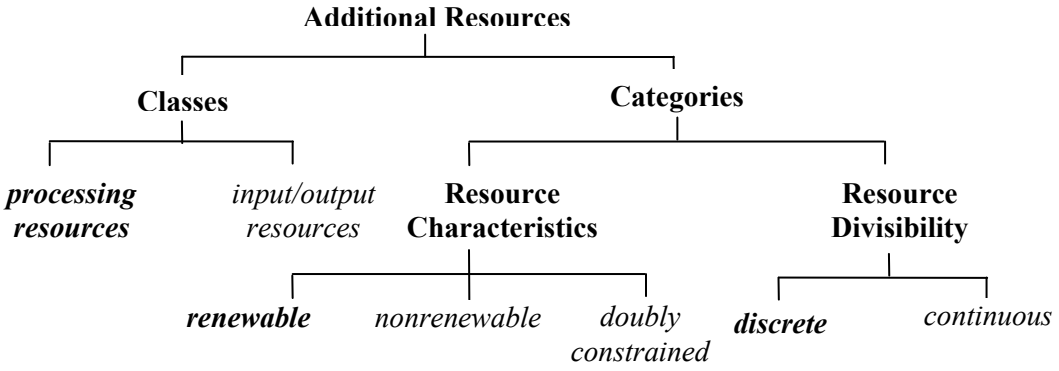


Figure 3.1 Classification of additional resources (Blazewicz, Brauner & Finke, 2004; Blazewicz et al. 2007b; Slowinski, 1980)

Since the proposed research in this dissertation addresses non-preemptive jobs without precedence constraints, the studies including precedence constraints and preemptive tasks are also not reviewed here. Interested readers on these fields are referred to Blazewicz, Cellary, Slowinski, & Weglarz (1986) and Blazewicz et al. (2007b).

Unless explicitly indicated, throughout this chapter, we assume that:

i. A job cannot be processed on more than one machine simultaneously.
ii. A machine cannot process more than one job at a time.
iii. No precedence constraints are allowed.
iv. Preemption is not allowed.
v. Job cancellation is not allowed.
vi. Processing times are independent of the schedule.
vii. Machines are always available.
viii. Jobs are all known in advance.
ix. The problem is purely deterministic.

Throughout this chapter, the classification scheme presented in Chapter 2 is used. The studies related to RCPMSPs in the literature are summarized in Appendix A.

In this review chapter, the studies are evaluated in five main topics:

a. Machine Environment Characteristics
b. Additional Resource Characteristics
c. Objective Functions
d. Solution Methods
e. Other Important Issues

The following sub-sections analyze the related studies by focusing their strengths as well as weaknesses on these five main topics, consecutively.

3.2 Machine Environment Characteristics

The analysis of surveyed papers in this sub-section is done in two aspects.

- the number of machines considered, and
- the characteristics of the machine environment

In terms of the number of machines, most of the studies except a few deal with more than three machines. As expected; almost all papers concerning two or three machines either prove NP-hardness of the investigated problems or propose exact algorithms to reach optimal solutions (e.g., Blazewicz, Lenstra & Rinnooy Kan, 1983; Blazewicz, Barcelo, Kubiak, & Rock, 1986; Blazewicz, Kubiak, Röck, & Szwarcfiter, 1987; Garey & Johnson, 1975; Kellerer & Strusevisch, 2003; Kellerer & Strusevisch, 2004).

Recall that classical PMS theory classifies the machine environment into three main classes: identical, uniform and unrelated parallel machines. However, literature related to RCPMSPs investigates a new category, *parallel dedicated machines*. In this new category, the set of jobs that will be processed on each machine is pre-determined. Surely, this assumption simplifies the entire RCPMSP by eliminating job-machine assignment sub-problem. Almost half of the studies surveyed in this chapter assume that machines are dedicated.

Another widely studied machine environment is the case of identical parallel machines which eases the design and implementation of exact and/or approximation algorithms. For instance, Blazewicz, Lenstra & Rinnooy Kan (1983) and Ventura & Kim (2000) propose polynomial time exact algorithms for RCPMSPs with identical machines. Uniform (Kovalyov & Shafransky, 1998; Ruiz-Torres, Lopez & Ho, 2007) and unrelated machines (Grigoriev, Sviredenko & Uetz, 2005, 2006, 2007) are rarely studied.

On the other hand, as already stated in Chapter 2, machine eligibility restrictions may be viewed as a special case of unrelated parallel machine environment. In case of machine eligibility restrictions, job i is only allowed to be processed on a subset M_i of the m machines in parallel. To the best of our knowledge, no study, so far, takes machine eligibility restrictions into account for RCPMSPs. In another related field, PMS with auxiliary equipment of constraints, few of the studies (Chen, 2005; Chen & Wu, 2006; Tamaki, Hasegawa, Kozasa & Araki, 1993) consider machine eligibility restrictions. However, these studies consider only the dies as additional resources which may not be treated as a common shared resource (e.g., machine operators).

Relaxing the first assumption given in Section 3.1, we may allow one machine may process more than one job at a time, In this class of scheduling problems, a facility (or machine) has a fixed capacity and each job requires a specified amount of this capacity. Some of the studies (Chu & Xia, 2005; Hooker, 2005, 2006) assume that facilities (or machines) may process more than one job at a time. Nevertheless, absence of additional resource(s) other than the main resource (machine) takes us away from classifying these problems into RCPMSPs.

3.3 Additional Resource Characteristics

In a significant number of studies surveyed, job processing times are not-fixed but based on the amount of additional resource allocated to it. Daniels, Hoopes & Mazzola (1996) name this problem as parallel machine flexible resource scheduling

(PMFRS) problem. A common characteristic in this field is that all the studies deal with a single additional resource in limited supply. This type of problems should solve an additional *resource allocation* problem since the amount of additional resource allocated to a job determines its processing time. Kellerer & Strusevisch (2008) also name the additional resource in PMFRS problems as "renewable speeding-up resource" since if job i is not given the resource, its processing time remains equal to p_i; otherwise the additional resource will speed up the processing in proportional to the allocated amount of additional resource.

In PMFRS problems, resource allocation problem itself adds a significant complexity to the entire problem. Therefore, most of the studies in this field relax other characteristics of the problem.

An instance of these simplifications occurs in the assignment of jobs to manufacturing cells (or machines). PMFRS problems in its original form assume that assignment of jobs to machines is pre-specified (see Daniels, Hoopes & Mazzola, 1996, 1997; Grigoriev & Uetz, 2006, 2009; Grigoriev, Sviridenko & Uetz, 2005; Kellerer & Strusevisch, 2008; Olafsson & Shi, 2000; Ruiz-Torres, Lopez & Ho, 2007). This assumption eliminates the job-machine assignment sub-problem from the entire problem further simplifying the problem at hand. This situation matches the term of *parallel dedicated machines* which has been introduced in Chapter 2.

Another simplifying assumption in PMFRS problems arises with the restricted use of additional resources through the machines. A number of studies (Daniels, Hua & Webster, 1999; Ruiz-Torres & Centeno, 2007; Ruiz-Torres, Lopez & Ho, 2007; Sue & Lien, 2009) assume that the flexible resource may be allocated freely among the machines, but the resulting resource assignment must remain fixed for the whole scheduling horizon. Daniels, Hoopes & Mazzola (1996) named this problem as *static* PMFRS. As stated by Olafsson & Shi (2000), in this case, the resources are inevitably non-renewable and there is no competition for additional resource across the machines. In the static case with the makespan objective, there is no need to

sequence the jobs subject to resource constraints, and the problem reduces to a simple resource allocation problem (Olafsson & Shi, 2000).

Let R denote total amount of additional resource in the system at any point in time. Each job i can then be processed with a specified amount of additional resource available. Let $\hat{p}_{ir}$ denotes the processing time of job i when $r \le b$ units of resources are allocated. Assume that r_j denotes the amount of resource assigned to each machine j. Daniels, Hoopes & Mazzola (1996) give the mathematical formulation of the static PMFRS problem (with the makespan objective) as follows:

$$\min \ C_{\max}$$

subject to:

$$\sum_{i \in N_j} \sum_{r=0}^{R} \hat{p}_{ir} x_{ir} \le C_{\max} \qquad j = 1, \ldots, m \qquad (3.1)$$

$$\sum_{r=0}^{R} r \times x_{ir} \le r_j \qquad j = 1, \ldots, m,\ i \in N_j \qquad (3.2)$$

$$\sum_{r=0}^{R} x_{ir} = 1 \qquad i = 1, \ldots, n \qquad (3.3)$$

$$\sum_{j=1}^{m} r_j \le R \qquad (3.4)$$

$$r_j \ge 0 \qquad j = 1, \ldots, m \qquad (3.5)$$

$$x_{ir} \in \{0, 1\}; \qquad i = 1, \ldots, n\ ;\ r = 0, \ldots, R \qquad (3.6)$$

where decision variable $x_{ir} = 1$ if job i is processed with r units of additional resource, and $x_{ir} = 0$ otherwise.

Constraints (3.1) define the makespan. Constraints (3.2) ensure that the amount of resource assigned to each job should not exceed the amount of additional resource allocated to corresponding machine. Constraints (3.3) ensure that a fixed amount of additional resource is assigned to each job. Constraint (3.4) limits total amount of

resource allocated among machines. Finally, Constraints (3.5) and Constraint (3.6) define the domain of decision variables.

Daniels, Hoopes & Mazzola (1996) propose an algorithm which gives optimal solution to this static PMFRS problem. First, the minimum amount of resource is assigned to each machine so that each job can be processed with their slowest processing times. Next, the machine whose total processing load determines the current makespan is identified, say machine j. Then, if sufficient size of additional resource remains, the amount of resource assigned to machine j (r_j) is increased by one unit, and the amount of remaining resource is decreased accordingly. This process is repeated, until no amount of additional resource remains.

Daniels, Hua & Webster (1999) extend the below formulation for the case where job assignment to machines is unspecified. In other words, the assumption that "machines are dedicated" is relaxed. They name this problem as static unspecified PMFRS (UPMFRS) and state that UPMFRS problem is NP-hard. They analyzed the joint contribution of job assignment and resource flexibility issues. It should be noted that resource allocation still remains its static case (i.e., static UPMFRS) which means that the resulting resource assignment of machines remains fixed for the whole scheduling horizon and the set of jobs assigned to each machine can be processed in any sequence without affecting the system makespan. The static version of UPMFRS problem with identical machines can be formulated as follows (Daniels, Hua & Webster, 1999):

$$\min \; C_{\max}$$

subject to:

$$\sum_{i=1}^{n}\sum_{r=0}^{R} \hat{p}_{ir} x_{ir} y_{ij} \le C_{\max} \qquad j = 1, \ldots, m \qquad (3.7)$$

$$\sum_{j=1}^{m} y_{ij} = 1 \qquad i = 1, \ldots, n \qquad (3.8)$$

$$\sum_{r=0}^{R} r \times x_{ir} \le r_j \qquad j = 1, \ldots, m, \; i = 1, \ldots, n \qquad (3.9)$$

$$\sum_{r=0}^{R} x_{ir} = 1 \qquad i = 1, \ldots, n \qquad (3.10)$$

$$\sum_{j=1}^{m} r_j \le R \qquad (3.11)$$

$$x_{ir} \in \{0,1\} \qquad i = 1, \ldots, n \; ; r = 0, \ldots, R \qquad (3.12)$$

$$y_{ij} \in \{0,1\} \qquad i = 1, \ldots, n; j = 1, \ldots, m \qquad (3.13)$$

$$r_j \ge 0 \qquad j = 1, \ldots, m \qquad (3.14)$$

where decision variable $y_{ij} = 1$ if job i is assigned to machine j, and $y_{ij} = 0$ otherwise.

Notice that this formulation is nonlinear due to the presence of Constraints (3.7) which determine the makespan. Constraints (3.8) specify that each job is assigned to one and only one machine. The remaining constraints are similar to the ones of static PMFRS formulation given before. Note that (3.12) through (3.14) define the domains of the decision variables. Ruiz- Torres & Centeno (2007), Ruiz-Torres, Lopez & Ho (2007) and Sue & Lien (2009) also deal with static version of UPMFRS problem.

The study of dynamic case, where the flexible resource can switch between machines during the schedule is more realistic. The dynamic PMFRS problem includes three sub-problems. First, a dynamic resource allocation must be determined. Second, a sequence in which jobs should be processed within each machine must be determined, and a starting time for each job has to be found (Olafsson & Shi, 2000). Daniels, Hoopes & Mazzola (1996) also formulated the dynamic version of the PMFRS problem. The decision variables are defined as:

$$y_{ih} = \begin{cases} 1, & \text{if job } i \text{ precedes job } h, \text{ where } i, h \in N_j. \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{irt} = \begin{cases} 1, & \text{if job } i \text{ completes its processing with } r \text{ units of resource at time } t. \\ 0, & \text{otherwise.} \end{cases}$$

Let p_i denote the actual processing time of job i and T be an upper bound on the makespan. For example, $T = \max_{j \in M} \sum_{i \in N_j} \hat{p}_{i0}$. The formulation is as follows (Daniels, Hoopes & Mazzola, 1996; Olafsson & Shi, 2000):

$$\min \; C_{\max}$$

subject to:

$$\sum_{r=0}^{R} \hat{p}_{ir} \sum_{t=1}^{T} x_{irt} = p_i \qquad i = 1, \ldots, n \qquad (3.15)$$

$$\sum_{t=1}^{T} t \left(\sum_{r=0}^{R} x_{irt} \right) = C_i \qquad i = 1, \ldots, n \qquad (3.16)$$

$$\sum_{r=0}^{R} \sum_{t=1}^{T} x_{irt} = 1 \qquad i = 1, \ldots, n \qquad (3.17)$$

$$C_i \le C_{\max} \qquad i = 1, \ldots, n \qquad (3.18)$$

$$C_h - C_i + T(1 - y_{ih}) \ge p_h \qquad i \in N_j,\ h \in N_j \setminus \{i\},\ j = 1, \ldots, m \qquad (3.19)$$

$$y_{ih} + y_{hi} = 1 \qquad i \in N_j, h \in N_j \setminus \{1,2,\ldots,i\},\ j \in M \qquad (3.20)$$

$$\sum_{i=1}^{n} \sum_{r=0}^{R} \sum_{l=t}^{t+\hat{p}_{ir}-1} r \times x_{irl} \le R \qquad t = 1, \ldots, T \qquad (3.21)$$

$$x_{irt} \in \{0,1\} \qquad i = 1,\ldots,n\ ;\ r = 0,\ldots,R,\ t = 1, \ldots, T \qquad (3.22)$$

$$y_{ih} \in \{0,1\} \qquad i \in N_j,\ h \in N_j \setminus \{i\},\ j \in M \qquad (3.23)$$

$$C_i \ge 0 \qquad i = 1, \ldots, n \qquad (3.24)$$

Constraints (3.15) and (3.16) determine the actual processing time and completion time of each job, respectively. Constraints (3.17) ensure that each job is assigned to fixed number of additional resource and accordingly completed on a unique time. Constraints (3.18) determine the makespan. Constraints (3.19) and (3.20) ensure that, within each machine j, either job $i \in N_j$ precedes job $h \in N_j$, or job h precedes job i. Constraints (3.21) state that total amount of additional resource consumed at each time should not exceed the total amount available. Finally, (3.22) through (3.24) define the domain of the decision variables.

Daniels, Hoopes & Mazzola (1996, 1997) compare the optimal solutions for the static and dynamic versions of the PMFRS problems with respect to makespan objective. They report significant improvements in the makespan performance in case a flexible resource is dynamically allocated across the machines. They also state that potential improvement on the makespan increases as the number of machines (m) and the sensitivity of job processing times to the allocation of additional resource (α) increase, and as the total amount of available resource (R) decreases. This observation suggests that resource flexibility is most beneficial in larger problems with relatively scarce resources, and when job processing times vary inversely with the amount of allocated resource.

Several studies (Grigoriev, Sviridenko & Uetz, 2005; Grigoriev & Uetz, 2009; Kellerer & Strusevisch, 2008; Olafsson & Shi, 2000) deal with dynamic cases of PMFRS problem.

To define the dynamic PMFRS problem in $\alpha|\beta|\gamma$ classification scheme, Kellerer & Strusevisch (2008) described new symbols, i.e., *Bi*, *Int*, *Lin*, to be used in the β field all of which refers that the processing times are resource-dependent (i.e., the problem is a PMFRS). More specifically (Kellerer & Strusevisch, 2008):

- "*Bi*" refers binary scenario of resource allocation, i.e., if job i is not given the resource, its processing time remains equal to p_i, otherwise; if job i is given the resource, its processing time becomes p_i - π_i. Exactly one unit of the resource required at any time of this processing.

- "*Int*" stands for integer scenario of resource allocation, i.e., if job $i \in N$ is given τ units of the resource, then its actual processing time is equal to a given $p_{i\tau}$. Exactly τ units of the resource required at any time of this processing.

- "*Lin*" stresses that the actual processing times depend linearly on the number of units of speeding-up resource allocated to the job; if job i is given τ units of the resource, then its actual processing time is equal to $p_{i\tau} = p_i - \tau \times \pi_i$.

In terms of classification defined above, Kellerer & Strusevisch (2008) define the dynamic PMFRS problem with binary resource allocation (i.e., 0/1 resource requirements) as $PDm|res111, Bi|C_{\max}$. In this representation, PDm refers to the case of m parallel dedicated machines which is a common characteristic of the PMFRS problem, whereas "$res111$" denotes that there is one additional resource type with one unit available and each job is allocated no more than one unit of additional resource. Finally, "Bi" in the middle field stresses that the resource is speeding-up and binary scenario of its consumption is applied.

Recall that, dynamic PMFRS still assumes the set of jobs to be processed on each machine is pre-specified. More general version of this problem is a UPMFRS problem with dynamic case which also includes job-machine assignment sub-problem. A number of studies (Grigoriev, Sviridenko & Uetz, 2005, 2006, 2007; Kellerer, 2008; Kellerer & Strusevisch, 2008) deal with dynamic version of UPMFRS problem and propose approximation algorithms.

It should be remembered that all of the studies related to PMFRS problems assume that there is only one additional resource type in the system.

Daniels, Hoopes & Mazzola (1996) first use the term of RCPMSP and classified it as a special case of the dynamic PMFRS problem in which actual processing times p_i and resource requirements r_i are given for each job i. However, remember that PMFRS problem in its original form, assumes that machines are dedicated and there is only one additional resource in limited supply. Therefore, RCPMSP, in our opinion, may cover a more general field since it may also consider job-machine assignment sub-problem and may allow the presence of more than one additional resource type. It only excludes resource allocation problem, and accordingly resource dependent processing times. In the light of this discussion, RCPMSP seems to be closer to dynamic UPMFRS problem which incorporates also job-machine assignment sub-problem.

Kellerer & Strusevisch (2008) also discuss the relations between RCPMSPs and PMFRS problems. They refer to resource type(s) in RCPMSPs as the "renewable static resource", since resource allocations are known in advance and accordingly processing times are also known in a priori.

A general RCPMSP with identical machines and k number of additional resource types can then be formulated as follows:

$res_{i,k}$ the amount of additional resource k required by job i.

b_k available amount of additional resource k.

u number of additional resource types

$$x_{it} = \begin{cases} 1, & \text{if job } i \text{ begins its processing at time } t. \\ 0, & \text{otherwise.} \end{cases}$$

$$\min \; C_{\max}$$

subject to:

$$\sum_{j \in M_i} \sum_{t=0}^{T-p_i+1} x_{it}(t + p_i) \le C_{\max} \qquad i = 1, \ldots, n \tag{3.25}$$

$$\sum_{i=1}^{n} \sum_{s=\max\{0,t-p_i\}}^{t-1} x_{is} \le m \qquad t = 0, \ldots, T \tag{3.26}$$

$$\sum_{t=0}^{T-1} x_{it} = 1 \qquad i = 1, \ldots, n \tag{3.27}$$

$$\sum_{i=1}^{n} \sum_{s=\max\{0,t-p_i\}}^{t-1} res_{i,k} x_{is} \le b_k \qquad k = 1, \ldots, u; \; t = 1, \ldots, T \tag{3.28}$$

$$x_{it} \in \{0, 1\} \qquad i = 1, \ldots, n \; ; \; t = 0, \ldots, T\text{-}1 \tag{3.29}$$

This IP model aims to minimize makespan working with Constraints (3.25). Constraints (3.26) make sure that no more than one job can be assigned to any machine at any time period. Constraints (3.27) ensure that each job should certainly be processed. Constraints (3.28) state that, for each additional resource type, total amount assigned to jobs at any time period is less than or equal to the available amount of each resource type, b_k. Finally, (3.29) ensures that all x_{it} are 0-1 variables.

Surely, the complexity of RCPMSPs grows with the number of additional resource types. Therefore, most of the studies relevant to RCPMSPs consider a single additional resource type. Moreover, the studies that deal with more than one additional resource type consider

- unit (equal) processing times (Blazewicz, Lenstra & Rinnooy Kan, 1983; Blazewicz, Barcelo, Kubiak, & Rock, 1986; Blazewicz, Kubiak & Martello, 1993; Garey & Johnson, 1975; Srivastav & Stangier, 1997; Ventura & Kim, 2003) and/or
- two or three machines (Garey & Johnson, 1975; Blazewicz, Kubiak & Martello, 1993), or
- unit size of additional resources and only 0/1 resource requirements (Kellerer & Strusevisch, 2004).

Thus, other versions of RCPMSPs with more than one additional resource type still remain a potential research area.

To clarify the problem types studied in the surveyed papers, let us represent the problem characteristics that determine the type of the problems. Four main characteristics each of which has two sub-levels, indeed, determine the type of the problem:

- Number of additional resource types
    - One additional resource type (*one*)
    - One or more additional resource type(s) (*one*/*multi*)
- Effect of the additional resources on the processing types
    - Processing time reduces with the increasing amount of additional resources (*speeding-up resources*)
    - Resource requirements of jobs are fixed and known priori (*fix*)
- Allocation of resources
    - Each machine can only use a fixed amount of static additional resource i.e., the additional resource cannot switch across other machines during the schedule (*static*)

- The additional (flexible) resource(s) can switch between machines during the schedule (*dynamic*)

- Job-machine assignment
  - Assignment of jobs to machines is pre-specified (*specified-dedicated*)
  - Assignment of jobs to machines is unspecified (*unspecified*)

Figure 3.2 illustrates the six main problem types with respect to these four problem characteristics defined above. Note that, for illustration purposes, the phrases written in parentheses above are used in Figure 3.2.

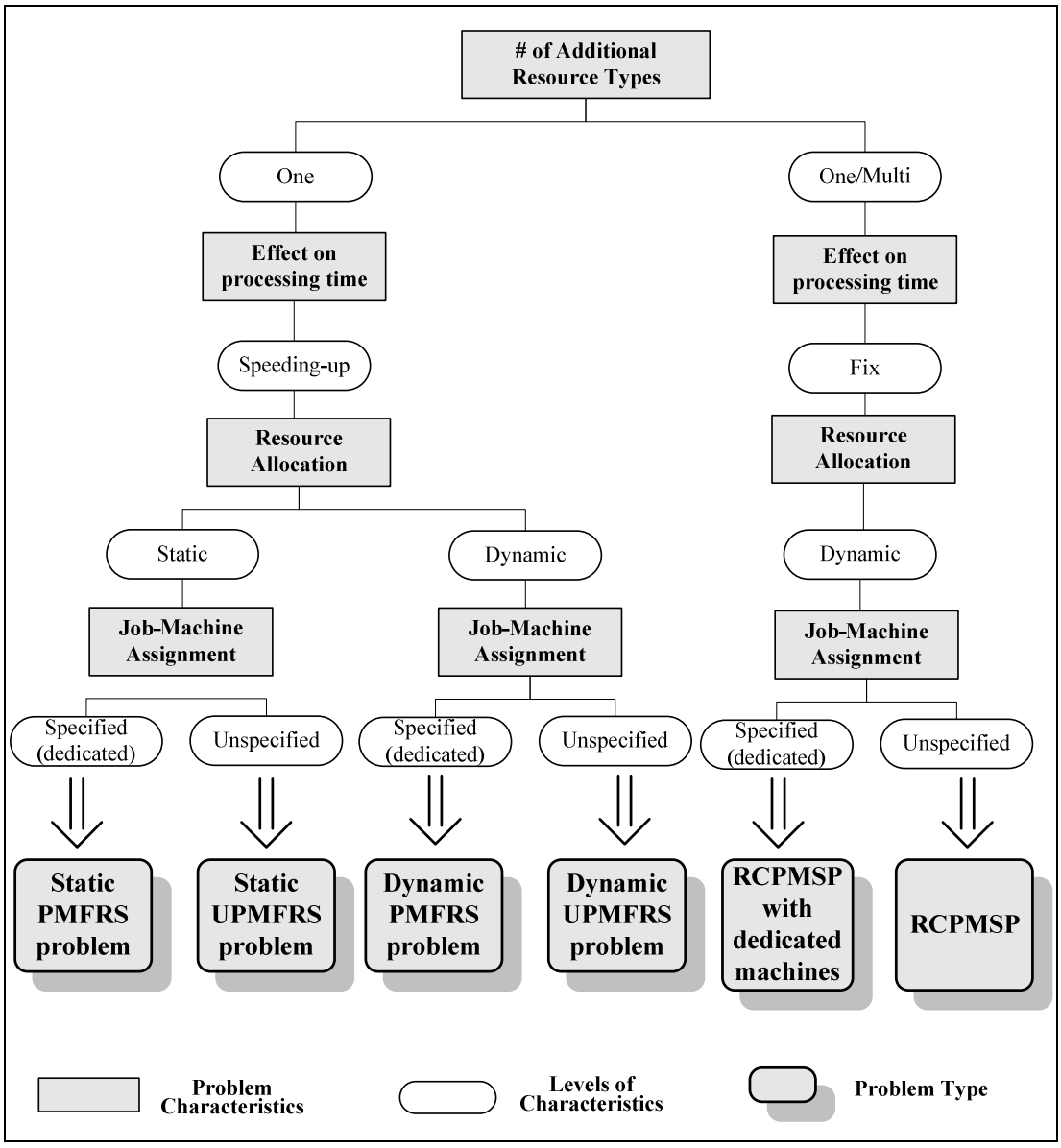


Figure 3.2 Characteristics of the problem types

Each problem type is defined by the characteristics on its own way in the tree. For instance, a static UPMFRS problem has one additional speeding-up resource type with static resource allocation and unspecified job-machine assignment.

Finally, in all problem types discussed above, it is assumed that additional resource(s) are allowed to be commonly used by all jobs. Rather than common shared case, different use of additional resources may occur in the manufacturing environments. For instance, Tamaki et al. (1993), Chen (2005) and Chen & Wu (2006) handle an unrelated parallel machine problem with a secondary resource constraint, e.g., dies. These papers assume that only the number of auxiliary equipment (e.g., dies) is limited, instead of considering a common shared resource (e.g., machine operators).

3.4 Objective Function(s)

Excluding a few papers, all the studies surveyed in this chapter aim to minimize makespan. It may be due to a number of reasons:

- The makespan objective is widely used in PMS studies; since it balances the load between machines and accordingly provides a high utilization of the machines (Pinedo, 1995).

- It is easier to handle the makespan objective when compared to other criteria, such as due date based objectives.

- Surely, classical PMS problems are relaxed versions of the ones with additional resource constraints. There exist a number of efficient solution algorithms to minimize makespan in classical PMS literature. Therefore, these solution algorithms are utilized in designing approximation approaches for RCPMSPs.

- Developing lower bounds for the makespan objective is rather straightforward in comparison to other performance criteria.

All but one of the studies related to PMFRS or UPMFRS problems deal with makespan performance criterion. Only Ruiz-Torres, Lopez & Ho (2007) aim to minimize the number of tardy jobs for the static UPMFRS problem with uniform machines.

The other objective functions considered in the related literature are minimizing lateness (Blazewicz, Barcelo, Kubiak & Röck 1986; Blazewicz, Kubiak & Martello, 1993); minimizing total absolute deviation either from common due dates, i.e., *TAD* (Ventura & Kim, 2000) or from arbitrary due dates i.e., *TADD*, (Ventura & Kim, 2003) and minimizing total flow time (Blazewicz et al., 1987).

An interesting point is that all papers deal with a single objective function. The complex nature of this class of problems has prevented development of models with multiple criteria. The problems with multiple criteria may be handled by applying either sequential optimization procedures or simultaneous optimization approaches with appropriate weights.

3.5 Solution Methods

Solution approaches related to RCPMSPs spread out over a wide area from polynomial time exact approaches to meta-heuristics. Figure 3.3 illustrates the solution approaches applied in the surveyed papers.

A number of studies give polynomial-time algorithms for some special cases of the problems. However, these studies handle simple cases with two or three machines, 0/1 resource requirements, or dedicated machine environments. Several studies, on the other hand, propose problem-based heuristics, approximation algorithms and meta-heuristic approaches. Throughout this section, the details of these solution approaches will be given.

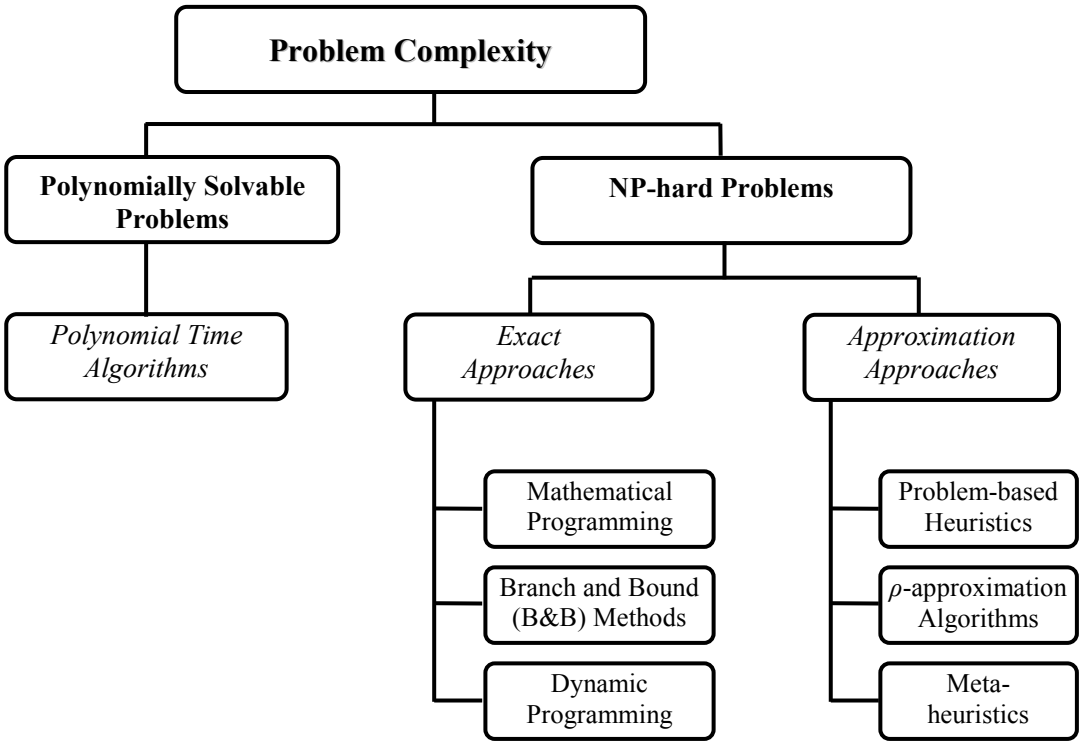


Figure 3.3 Classification of solution approaches

3.5.1 Polynomially Solvable Problems

Recall that almost all PMS problems with additional resources are NP-hard (Blazewicz, Lenstra & Rinnooy Kan, 1983). However, for a number of specialized cases, researchers have developed polynomial time exact algorithms. Table 3.1 lists these problems in the chronological order with complexity functions of proposed algorithms.

For the problem of $P2|res\cdots, p_i=1|C_{\max}$, Garey & Johnson (1975) propose a polynomial-time algorithm with an idea of establishing the correspondence between maximum matching in a graph displaying resource constraints and the minimum length schedule. Blazewicz (1979) developed a polynomial time solution procedure for $P|res1\cdot 1, r_i, d_i|\varnothing$ problem where release times and deadlines of all jobs should be met. The proposed procedure first determines the modified deadlines, d_i^* by considering deadlines and release times of all jobs and then, by a constructive algorithm, schedules the jobs so that each job meets its modified deadline.

Table 3.1 Polynomially solvable RCPMSPs

Reference	**Year**	**Problem Classification**	**Algorithm Complexity**
Garey & Johnson	1975	$P2\|res\cdots, p_i=1\|C_{\max}$	$O(n^3)$
Blazewicz	1979	$P\|res1\cdot 1, r_i, d_i\|\varnothing$	$O(n^2)$
Blazewicz, Lenstra & Rinnooy Kan	1983	$Q2\|res1\cdot\cdot, p_i=1\|C_{\max}$	$O(n\log n)$
Blazewicz & Ecker	1983	$P\|res\lambda\sigma\delta, p_i=1\| C_{max}$	$O(\log n)$
Blazewicz et al.	1987	$P\|res1\cdot 1\|\sum_i C_i$	$O(n^3)$
Daniels, Hoopes & Mazzola	1996	Static PMFRS	$O(nR\ (n+m))$
Kovalyov & Shafransky	1998	$P\|res1\cdot 1, p_i=1\|C_{\max}$	$O(1)$
Kovalyov & Shafransky	1998	$Q\|res1\cdot 1, p_i=1, nmit\|C_{\max}$	$O(m\log m)$
Ventura & Kim	2000	$P\|res1\cdot 1, p_i, d_i=d\|\ TAD$	Polynomial-time
Ventura & Kim	2003	$P\|res1\cdot 1, r_i,\ p_i=1, d_i\|\ TADD$	Polynomial-time
Kellerer & Strusevisch	2003	$PD2\|res111\|C_{\max}$	$O(n)$
Kellerer & Strusevisch	2004	$PD2\|res1\cdot\cdot\|C_{\max}$	$O(n\log n) + O(n)$
Kellerer & Strusevisch	2004	$PD2\|res211\|C_{\max}$	$O(n)$

Blazewicz, Lenstra & Rinnooy Kan (1983) also give a polynomial time algorithm for solving $Q2|res1\cdot\cdot, p_i=1|C_{\max}$ problem optimally. Suppose that speed of the first machine is greater than or equal to the speed of the second one ($v_1 \geq v_2$). The algorithm, at first, schedules all jobs on the first machine in non-increasing order of resource requirement. Next, it consecutively removes the last job from the first machine and schedules it as early as possible on the second machine, as long as the value of $C_{\max}$ is reduced. Blazewicz & Ecker (1983) state that a generalized version of this problem, i.e., $P|res\lambda\sigma\delta, p_i=1|\ C_{max}$, may be identified as determining a partition of job set M into concurrently executable subsets where the number of subsets is to be minimized. Therefore, they state that this problem is equivalent to λ-dimensional, σ-restricted bin packing problem which can be solved in O (log n) time. Kovalyov & Shafransky (1998) present an O($m$log$m$) algorithm for the $Q|res1\cdot 1, p_i=1, nmit|C_{\max}$ problem where the case of no machine idle times (*nmit*) ensures that no machine may stand idle, unless all jobs allocated to this machine have completed their processing. The idea behind the algorithm is that all resource jobs (i.e., jobs requiring additional resource) are assigned to first fastest σ machines

while the remaining non-resource jobs are allocated to ($m - \sigma$) slow machines. Recall that σ is part of the input and denotes the size of the additional resource. If the resulting makespan of the schedule on the fast machines is greater than or equal to the resulting makespan of the schedule on slow machines, combination of these two schedules gives the optimal schedule; otherwise, a different algorithm that considers all jobs together is applied to obtain the optimal schedule. Kovalyov & Shafransky (1998) also state that the given algorithm can be extended to other objective functions of the same problem. Finally, they state that, for the identical machine case, a modification of the proposed algorithm gives optimal solution in O(1) time complexity.

For the problems with identical machines, one additional resource type and 0/1 resource requirements, an optimal schedule exists in case that all jobs requiring one unit of additional resource are assigned to the first b (where $b \leq m$) machines, where the additional resource is available in b units (see Blazewicz et al., 1987; Ventura & Kim, 2000, 2003).

As already given in Section 3.3, Daniels, Hoopes & Mazzola (1996) propose a polynomial time algorithm to static PMFRS problem.

Kellerer & Strusevisch (2003, 2004) develop polynomial time algorithms for various versions of problems with two parallel dedicated machines. For $PD2|res111|C_{\max}$, Kellerer & Strusevisch (2003) develop a group technology-based polynomial algorithm where each subset of jobs assigned to each machine j (i.e., N_j) is split into further two subsets: the subset of non-resource jobs Q_j, and the subset of resource jobs R_j. All jobs of these subsets are considered as a single batch. Notice that, the batches composed of non-resource jobs may overlap with any other batch; while the batches composed of resource jobs cannot be processed simultaneously within each other. The algorithm assigns the batch composed of R_1 and then the one of Q_1 to the first machine consecutively without intermediate idle time. Then, it starts batch of Q_2 on the second machine at time zero, and finally it starts batch of R_2 on the second machine as early as possible. The resulting makespan is optimal. Kellerer

& Strusevisch (2004) also develop polynomial time algorithms for two extensions of this problem. First, for the problem where the resource size and requirements are arbitrary, i.e., $PD2|res1\cdot\cdot|C_{\max}$, they develop a polynomial-time algorithm by using machine and resource based lower bounds. They first partition the subset of jobs for each machine N_j into further (σ+1) subsets where each subset contains all jobs with the same resource requirements (r = 0, 1, …, σ). Then, subsets of the first machine, beginning from the ones with no-resource requirements (r = 0) and ending with the ones with σ resource requirements are allocated, successively. Accordingly, subsets of the second machine are assigned in the reverse order. Finally, utilizing the derived lower bounds, start times of the subsets are modified to eliminate the overlaps. In the same paper, they extend the $PD2|res111|C_{\max}$ problem discussed in Kellerer & Strusevisch (2003) to include two additional resource types. Resource-based lower bounds in this case are calculated based on subsets of resource and non-resource jobs with respect to both two additional resource types. Then the subsets are ordered in a way that will give the optimal schedule.

Consequently, a considerable amount of research related to RCPMSPs has been devoted to finding efficient polynomial time algorithms. However, many problems, yet, do not have a polynomial time algorithm; these problems are the so-called NP-hard problems. The following section presents and discusses the related NP-hard problems.

3.5.2 NP-hard Problems Proved in the Literature

Other than the problems listed in Table 3.1, almost all cases related to RCPMSPs are NP-hard (Blazewicz, Lenstra & Rinnooy Kan, 1983). Table 3.2 gives the NP-hard problems proved in the literature. Surely, according to complexity hierarchy given in Section 2.3.4 and reductions between types of resource constraints given in Section 2.5, other more complex versions of RCPMSPs are also NP-hard. For instance, since $Q2|res1\cdot\cdot, p_i=1|C_{\max}$ problem is NP-hard, $Q2|res\cdot 11, p_i=1|C_{\max}$ and $Q2|res1\cdot\cdot, p_i=1|L_{\max}$ are also NP-hard.

Let us discuss the parameters that have an influence on the hardness of the problem. First, different ready times cause strong NP-hardness of the problem. Although $P2|res\cdots,\ p_i=1|C_{\max}$ can be solved in polynomial time (see Table 3.1), $P2|res1\cdot\cdot,\ r_i,\ p_i=1|\ C_{\max}$ (even with the presence of one additional resource type) is strongly NP-hard. Second, an increase of the number of machines from two to three causes a strong NP-hardness of the problem. For instance, while the problem of $P2|res1\cdot\cdot,\ p_i=1|C_{\max}$ can be solved in polynomial time, $P3|res1\cdot\cdot,\ p_i=1|C_{\max}$ problem is NP-hard.

Table 3.2 NP-hard problems proved in the literature

Reference	**Year**	**Problem Classification**
Garey & Johnson	1975	$P3\|res1\cdot\cdot,\ r_i,\ p_i=1\|C_{\max}$
Blazewicz, Lenstra & Rinnooy Kan	1983	$P3\|res\cdot11,\ p_i=1\|C_{\max}$
Blazewicz, Lenstra & Rinnooy Kan	1983	$Q2\|res\cdot11,\ p_i=1\|C_{\max}$
Blazewicz , Barcelo, Kubiak, & Rock	1986	$P2\|res1\cdot\cdot,\ p_i=1\|L_{\max}$
Blazewicz , Barcelo, Kubiak, & Rock	1986	$P2\|res\cdot11,\ p_i=1\ \|\ L_{\max}$
Blazewicz, Cellary, Slowinski, & Weglarz	1986	$P2\|res1\cdot\cdot,\ r_i,\ p_i=1\|\ C_{max}$
Blazewicz et al.	1987	$P2\|res1\cdot\cdot\|\sum_i C_i$
Blazewicz et al.	1987	$P2\|res\cdot\cdot1\|\ \sum_i C_i$
Kellerer & Strusevisch	2003	$PD3\|res111\|C_{\max}$
Kellerer &Strusevisch	2003	$PDm\|res111\|C_{\max}$
Kellerer &Strusevisch	2004	$PD2\|res222\|C_{\max}$
Kellerer &Strusevisch	2004	$PD2\|res311\|C_{\max}$
Kellerer &Strusevisch	2008	$PD2\|res111, Bi\|C_{\max}$ (PMFRS)

In the view of solution methods for solving NP-hard RCPMSPs, the investigation focuses on two aspects:

- Optimally solving special, more tractable cases with exact methods (e.g., B&B methods, dynamic programming)
- Developing approximation algorithms (e.g., heuristics and metaheuristics)

Table 3.3 presents taxonomy of the exact and approximation solution procedures with respect to each study. The following two sub-sections, on the other hand, review these studies based on specific solution methods by focusing on above two aspects.

3.5.3 Exact Methods

Exact algorithms in the literature are relatively few. Blazewicz, Kubiak & Martello (1993) implement a depth first B&B algorithm for the $P2|res\cdots, p_i=1|L_{\max}$ problem. Jobs are ordered according to the earliest due date (EDD) rule and decision nodes are generated by assigning a job to one of the two machines step by step. A significant number of decision node explorations are eliminated by embedding the investigated lower bounds and dominance relations into each node. This B&B algorithm was able to produce optimal results up to 1000 jobs for one additional resource type, and up to 100 jobs for more than one additional resource type.

Daniels, Hoopes & Mazzola (1996) develop a B&B algorithm for the dynamic PMFRS problem. Since the amount of the additional resource allocated to a job determines its processing time in PMFRS problems, they created a sub-problem series of RCPMSPs, in which, processing time p_i and resource requirements r_i are specified. For each sub-problem, lower bounds on makespan are calculated and resource-allocation policies that cannot give better solutions are fathomed. For each candidate resource-allocation policy, the algorithm proceeds a lower level search where job sequencing and start time decisions of jobs are considered simultaneously. This B&B algorithm is able to solve the problems with up to 15 jobs and four machines. Ruiz-Torres, Lopez & Ho (2007) present a simple IP formulation to be used in solving static PMFRS problem with the objective of minimizing number of tardy jobs when the problem is small. In a recent study, Kellerer & Strusevisch (2008) propose a dynamic programming algorithm for $PD2|res1\cdot\cdot, Bi|C_{\max}$ problem (i.e., a dynamic PMFRS problem with binary resource allocation). The dynamic programming algorithm solves two knapsack sub-problems formulated for each machine.

Table 3.3 A taxonomy of problems based on exact and approximation solution procedures

Reference	Year	Problem Classification	Solution Method
Blazewicz, Kubiak& Martello	1993	$P2\|res\cdots, p_i=1\| L_{\max}$	Problem-based heuristic algorithms
		$P2\|res\cdots, p_i=1\| L_{\max}$	A B&B algorithm
Daniels, Hoopes & Mazzola	1996	Dynamic PMFRS	A B&B algorithm
			A static-based heuristic (SBH)
Daniels, Hoopes & Mazzola	1997	Dynamic PMFRS	A tabu search heuristic
			A static-based tabu search heuristic
Daniels, Hua & Webster	1999	Static UPMFRS (identical machines)	A decomposition heuristic (using SBH)
			A TS heuristic based on static-based heuristic
Grigoriev & Uetz	2009	Dynamic PMFRS	A (nonlinear) integer mathematical program-based (3+ε)-approximation algorithm
Grigoriev, Sviridenko & Uetz	2005	Dynamic UPMFRS (unrelated machines)	Based on a rounding algorithm for the relaxed formulation, a (4+2√2)-approximation algorithm
		Dynamic PMFRS	Based on a rounding algorithm for the relaxed formulation, a (3+2√2)-approximation algorithm
Grigoriev, Sviridenko &Uetz	2006	Dynamic UPMFRS (unrelated machines)	Based on a LP rounding algorithm for the relaxed formulation, a 3.75-approximation algorithm
Grigoriev, Sviridenko &Uetz	2007	Dynamic UPMFRS (unrelated machines)	Based on a LP rounding algorithm for the relaxed model, a 4-approximation LP-Greedy algorithm
Kellerer	2008	Dynamic UPMFRS (unrelated machines)	(3.5+ε) approximation algorithm
Kellerer & Strusevisch	2003	$PDm\|res111\|C_{\max}$	O(mn) Group Technology approximation alg.
		$PD3\|res111\|C_{\max}$	O(n) heuristic approximation algorithm
		$PD4\|res111\|C_{\max}$	O(n) heuristic approximation algorithm
		$PDm\|res111\|C_{\max}$	PTAS
Kellerer & Strusevisch	2004	$PD\|res\lambda 11\|C_{\max}$	An O(nm min $\{n,m\}$) greedy approximation alg.
		$PDm\|res\lambda 11\|C_{\max}$	PTAS
Kellerer & Strusevisch	2008	$PD2\|res111, Bi\|C_{\max}$	A pseudo-polynomial time dynamic prog. alg.
		$PD2\|res111, Bi\|C_{\max}$	FPTAS
		$PD\|res111, Bi\|C_{\max}$	Polynomial time (3/2)-approximation algorithm
		$PD\|res1\sigma\sigma, Int\|C_{\max}$	Polynomial time (3+ε)-approximation algorithm
		$PDm\|res111, Bi\|C_{\max}$	PTAS
Krause, Shen & Schwetman	1973	$P\|res1\cdot\cdot, p_i=1\|C_{\max}$	Polynomial-time (4/3)-approximation algorithm
Li, Wang & Lim	2003	$PD\|res1\cdot\cdot\|C_{\max}$	A genetic algorithm (GA)
Olafsson & Shi	2000	Dynamic PMFRS	A heuristic, named Nested Partitions (NP)
Ruiz Torres & Centeno	2007	Static UPMFRS (identical machines)	Problem-based heuristic algorithms
Ruiz-Torres, Lopez & Ho	2007	Static PMFRS	An mathematical programming formulation
		Static UPMFRS (uniform machines)	Five different problem-based heuristic algorithms
Srivastav & Stangier	1997	$P\|res\cdot 1, r_i, p_i=1 \mid C_{\max}$	A polynomial time approximation algorithm
Sue & Lien	2009	Static UPMFRS (identical machines)	Problem-based heuristic algorithms
Ventura & Kim	2003	$P\|res\cdots, r_i, p_i=1, d_i\|TADD$	Lagrangian-based heuristic algorithm

3.5.4 Approximation Algorithms

Given that many of RCPMSPs are NP-hard, it becomes necessary to work out approximation algorithms. In the scheduling literature, evaluation of approximation approaches is generally done by two ways. The first way is to evaluate their performance by calculating the worst-case bounds of the algorithms in a theoretical manner. The second way is to make experimental studies of proposed algorithms through a series of test problems and find the mean performances with respect to each other, or the lower bounds developed or optimal solutions (if any).

In the literature, there exist three well-defined simple heuristic rules: (see Blazewicz, Brauner & Finke, 2004; Blazewicz et al., 2007b, Eiselt & Sandblom, 2004):

First fit (FF). Each task is assigned to the earliest time slot in such a way that no resource or machine limits are violated.

First fit decreasing (FFD). A variant of the first algorithm applied to a list ordered in non-increasing order of *maximal relative resource requirement* $R_{\max}(i)$, where $R_{\max}(i) = \max \{ R_k(i) / b_k : 1 \le k \le u \}$.

Iterated lowest fit decreasing (*ILFD*: *for only* $u = 1$, *and* $p_i = 1$). Order the tasks as in the FFD algorithm. Put C as a lower bound on $C^*_{\max}$. Place job i in the first time slot and proceed the other tasks through the list by placing each job in a time slot for which total resource requirement of tasks already assigned is minimum. In case that job i cannot be assigned to any of the C slots, halt the iteration, increase C by 1, and start over.

Garey & Graham (1975) have shown that the performance bound of FF heuristic for the problem $P|res\cdots|C_{max}$ is:

$$R^{\infty}_{FF} = \min\left\{\frac{m+1}{2}, u+2-\frac{2u+1}{m}\right\}$$

Krause, Shen & Schwetman (1975) established performance ratios of above three heuristic methods, based on the problem, $P|res1\cdot\cdot, p_i=1|\ C_{max}$:

$$\frac{27}{10}-\left\lceil\frac{37}{10m}\right\rceil < R_{FF}^{\infty} < \frac{27}{10}-\frac{24}{10m},\ R_{FFD}^{\infty}=2-\frac{2}{m}, \text{ and } R_{ILFD}\le 2$$

Notice that, implementing *FFD* and *ILFD* algorithms which use ordered lists improves the performance by about 30%.

The following sub-sections give details of other solution approaches used in the related studies.

3.5.4.1 Problem-based Heuristic Algorithms

A number of papers in the literature develop problem-based heuristic algorithms. Blazewicz, Kubiak & Martello (1993) propose a basic greedy approach for the $P2|res\cdots,\ p_i=1|\ L_{\max}$ problem by specifying all job pairs that can be processed simultaneously subject to resource constraints. Ordering the jobs by EDD rule and breaking the ties by $\max_{1\le k\le u}\{R_k(i)/b_k\}$, they try to assign each job in the pairs to two machines consecutively. They also develop two improving heuristic algorithms to the greedy approach by introducing online and offline interchanges between jobs. We have already known from Table 3.1, Daniels, Hoopes & Mazzola (1996) developed a polynomial time exact algorithm for the static PMFRS problem. They also develop a heuristic algorithm, named static-based heuristic (SBH), for the dynamic PMFRS problem that repeatedly solves a series of static PMFRS sub-problems. Each sub-problem is solved to determine an optimal allocation of resources to machines by using the polynomial time algorithm of static PMFRS. The results show that the "static-based heuristic" works efficiently for the dynamic PMFRS problem with modest computational effort.

Daniels, Hua & Webster (1999) propose a decomposition heuristic for a static version of identical machine UPMFRS problem in which the assumption of predetermined job assignments is relaxed. In the decomposition algorithm, first, an

LPT rule is used to assign *n* jobs to *m* machines; and given this initial assignment, the polynomial-time static PMFRS algorithm of Daniels, Hoopes & Mazzola (1996) is applied to allocate the available resources to machines. The new processing time of jobs are calculated with respect to current resource allocation and again LPT rule is applied to find new job-machine assignments. This procedure goes on until a job-to-machine assignment encountered has been previously evaluated. The decomposition heuristic generates efficient results with modest computational effort. For the same static UPMFRS problem, Ruiz-Torres & Centeno (2007) develop a lower bound by full enumeration of resource assignment alternatives and propose new heuristic algorithms that combine list scheduling and bin packing approaches with rules to iteratively modify the flexible resource allocation. They demonstrate that their heuristic outperforms those in Daniels, Hua & Webster (1999) under most experimental combinations, and provides results that are close to the lower bounds. Ruiz-Torres, Lopez & Ho (2007) also propose five heuristic algorithms for static UPMFRS problem with the objective of minimizing number of tardy jobs. They developed "component heuristics" to make the job-machine assignments which is a part of "complete heuristics" that include also assignment of flexible resources to machines. They state that loading one machine at a time outperforms the approach of simultaneously loading all machines. They also analyzed the performance of the heuristics through different combination of problem parameters. Finally, in a recent study, Sue & Lien (2009) develop two heuristic algorithms for the same static UPMFRS problem. The former assigns jobs to machine first and then deploy the resources to jobs while the latter proceeds in the reverse manner. Computational results show that the latter heuristic dominates the former one.

Finally, it should also be noted that Lagrangian relaxation may also be a suitable technique for PMS problems with additional resources. By relaxing the resource constraints to the objective function with corresponding penalty costs, the remaining problem becomes probably easy to solve. Surely, in such a scheme, a problem-based heuristic is usually applied to convert the infeasible schedules of Lagrangian relaxation problem to the feasible ones. Ventura & Kim (2003) propose a Lagrangian-based heuristic algorithm for the $P|\, res \cdots, r_i, p_i{=}1, d_i|\, TADD$ problem by

relaxing the resource constraints and obtain efficient results for the test problems with up to 300 jobs, five machines and three additional resource types.

3.5.4.2 Approximation Algorithms with Worst-Case Bounds

As already stated, in the scheduling literature, evaluation of approximation approaches may also be done by calculating the worst-case bounds of the algorithms in a theoretical manner. Because of the strong-NP-hardness of the RCPMSPs, most researchers have designed approximation algorithms with increasingly better worst case ratios. A polynomial-time algorithm that creates a schedule with the objective function value that is at most $\rho \geq 1$ times the optimal value is called a $\rho - approximation$ algorithm; the value of ρ is called a *worst case ratio bound.*

In an earlier study, Krause, Shen & Schwetman (1973) proposed a 4/3-approximation algorithm for a multi-programmed computer system with identical task processors and restricted amount of memory which can be denoted as $P|res1\cdot\cdot, p_i=1|C_{\max}$. Srivastav & Stangier (1997) deal with a more general problem which is denoted as $P|res\cdot\cdot 1,\ r_i,\ p_i=1|C_{\max}$, and give polynomial time approximation algorithm with a worst case bound analysis in various cases. Kellerer & Strusevisch (2003) develop a polynomial time so-called "group technology approximation algorithm" for $PDm|res111|C_{\max}$ problem with a worst case bound of 3/2 for larger values of m. Moreover, for m=3 and m=4; i.e., $PD3|res111|C_{\max}$, and $PD4|res111|C_{\max}$, they propose polynomial time heuristic algorithms both of which have worst case bounds of 5/4. Kellerer & Strusevisch (2004) also develop a polynomial time greedy approximation algorithm for a problem with more than one additional resource type and arbitrary number of machines, i.e.; $PD|res\lambda 11|C_{\max}$. The worst case performance ratio of this algorithm is 2.

Recall that, a family of algorithms is called a *polynomial-time approximation scheme* (*PTAS*) if for a given ε>0 it contains an algorithm that has polynomial time running in the length of the program input and delivers a worst-case ratio bound of $1+\varepsilon$. Kellerer & Strusevisch (2003) propose a *PTAS* for the problem

$PDm|res111|C_{\max}$. For fixed m and $\varepsilon < 1$, the running time of the *PTAS* is polynomial in the size of the problem input but not in m and $1/\varepsilon$. Kellerer & Strusevisch (2004) extend their previous study by presenting a *PTAS* for $PDm|res\lambda 11|C_{\max}$ problem in which number of resources λ is also part of the input. Finally, in a recent paper, Kellerer & Strusevisch (2008) develop a *PTAS* for a simple dynamic PMFRS problem with binary resource allocation; i.e., $PDm|res111$, $Bi|C_{\max}$. They first try to "guess" a resource allocation that is very close to that in an optimal schedule, and then follow the lines of the PTAS obtained for the static problem given in Kellerer & Strusevisch (2003). Notice that, while the family of algorithms in a *PTAS* has a desirable effect that it can approximate arbitrarily closely to the optimal solution, it has an undesirable effect that its running time is not polynomial in $1/\varepsilon$. If additionally, the running time is polynomial in $1/\varepsilon$, a *PTAS* is called a fully polynomial-time approximation scheme (FPTAS). Kellerer & Strusevisch (2008) develop a *FPTAS* for a two-machine dynamic PMFRS problem with binary resource allocation, i.e., $PD2|res111$, $Bi|C_{\max}$. They formulated knapsack sub-problems for each of two machines, and used a rounding technique to solve these problems in polynomial time.

The recent studies in the literature show that a relaxed mathematical formulation of the original problem may be helpful in designing approximation algorithms. Grigoriev, Sviridenko & Uetz (2005) propose an approximation algorithm that gives $(4+2\sqrt{2})$ worst case bound for the dynamic UPMFRS problem with unrelated machines, i.e., the case that processing times are also machine dependent. The approximation method is based on an IP model that defines a relaxation of the original problem. The main idea is the utilization of an aggregate version of the resource constraints, yielding a formulation that does not require time-indexed variables. They then consider the LP relaxation of this relaxed formulation and apply a two-phase rounding procedure to assign resources to jobs, and jobs to machines. They finally apply a greedy list scheduling algorithm to generate a feasible schedule. They also adapted this method to the dynamic PMFRS problem and obtain a $(3+2\sqrt{2})$-approximation. They also show that LP-based analysis cannot yield anything better than a 2-approximation. Grigoriev, Sviridenko & Uetz (2006)

develop a better approximation algorithm for the same dynamic UPMFRS problem with unrelated machines. They apply a different and more efficient rounding algorithm to obtain resource allocations and job-machine assignments, and also a new scheduling algorithm inspired by the harmonic algorithm for bin packing to generate the final schedule. This new algorithm provides a 3.75-approximation for the problem at hand. Grigoriev, Sviridenko & Uetz (2007) review the proposed solution approaches given in their previous two studies described above and also show how to derive a 4-approximation algorithm using the relaxed formulation of the original problem and a simple list scheduling algorithm. They also show that the problem cannot be approximated within a factor smaller than 3/2.

Kellerer (2008) develops a $(3.5+\varepsilon)$-approximation algorithm for a more restricted UPMFRS problem with identical parallel machines. The solution procedure follows partially the approach of Grigoriev, Sviridenko & Uetz (2005) by transforming their IP formulation to a multiple choice knapsack problem. The resource allocation is done by solving this problem via a FPTAS. The jobs are then allocated to the machines using a variation of greedy list scheduling algorithm. The list scheduling algorithm works as follows: Among all non-assigned jobs assign the job with earliest possible starting time to the corresponding machine. If there is more than one job with earliest possible starting time, always choose the longest job among these jobs.

In a recent study, Grigoriev & Uetz (2009) propose a nonlinear-programming based $(3+\varepsilon)$-approximation algorithm for the dynamic PMFRS problem where the dependence of processing times on the amount of resources is linear for any job. The relaxed nonlinear programming formulation is solved approximately by a FPTAS and it gives the amount of resources allocated to every job. Then the jobs are scheduled according to an adaptation of the greedy list scheduling algorithm. The algorithm improves upon the 3.75-approximation from the Grigoriev, Sviridenko & Uetz (2007) and $(3.5+\varepsilon)$-approximation of Kellerer (2008) for the case of PMFRS problem. Moreover, the computation time of the new algorithm is polynomial in the input size and the precision $1/\varepsilon$.

Kellerer & Strusevisch (2008) develop a polynomial time 3/2-approximation algorithm for a PMFRS problem with a unit-size additional resource, binary resource requirements, and arbitrary number of machines, i.e., $PD|res111, Bi|C_{\max}$. They first apply a relaxed IP formulation by a FPTAS, which solves the resource allocation sub-problem and generates an instance of $PD|res111|C_{\max}$ problem. With solving this sub-problem they also found a lower bound on the makespan for the original problem. In the second phase, they applied a 3/2-approximation algorithm given in Kellerer & Strusevisch (2003). In a similar manner, Kellerer & Strusevisch (2008) also develop a polynomial time (3+ε)-approximation algorithm for more general version of PMFRS problem, which they denote as $PD|res1\sigma\sigma, Int|C_{\max}$. The algorithm proposed for this problem is also a two-phase procedure. In the first phase, the resource allocations are found by solving a relaxed quadratic IP problem by a FPTAS. And accordingly, in the second phase, the jobs are allocated to the machines using a simple greedy algorithm.

3.5.4.3 Metaheuristics

In the literature, metaheuristics (e.g., tabu search, simulated annealing, genetic algorithms) are widely applied to NP-hard scheduling problems. However, for the PMS problems with additional resources, the studies applying metaheuristics are rather a few. For the dynamic PMFRS problem, Daniels, Hoopes & Mazzola (1997) propose two tabu search heuristics, i.e., TSH, and TSH-SBH. Both heuristics employ a hierarchical search strategy based on the decomposition of three sub-problems of dynamic PMFRS problem; resource allocation, job sequence on each machine and job start times. Within this hierarchy, tabu search is applied to evaluate alternative resource allocation policies and job sequences. The first heuristic, named TSH, is applied on three initial solutions: each job is assigned the least possible amount of the resource, each job is assigned the maximum possible amount of the resource, and finally each machine is allocated an amount of the resource determined by static PMFRS algorithm of Daniels, Hoopes & Mazzola (1996). On the other hand, the second heuristic, named TSH-SBH, utilizes the final solution from the static-based heuristic as its only initial solution. Computational experiments state that TSH-SBH

produces more efficient solutions than individual TSH and SBH procedures. Daniels, Hua & Webster (1999) also propose a tabu search procedure for the static UPMFRS problem with identical machines. The procedure first determines the initial assignment of jobs by using LPT rule and applies the static PMFRS algorithm of Daniels, Hoopes & Mazzola (1996) to determine the allocation of resource to machines that minimizes makespan with the given job assignment. Alternative assignments are then generated by considering all pairwise exchange of jobs on different machines and evaluated by static PMFRS algorithm. If an improvement in makespan is observed, then the heuristic solution is updated and associated pairwise exchange is defined as a tabu move. The computational results show that tabu search procedure outperforms the decomposition heuristic developed by the same authors.

Olafsson & Shi (2000) propose a solution methodology, named Nested Partitions, which combines global sampling of the feasible region and local search heuristics. They first give an alternative formulation of dynamic PMFRS problem with the property of active schedules and prove that the optimal solution to the dynamic PMFRS problem is an active schedule. This property reduces the feasible region. Then, they apply the Nested Partitions method to the reformulated PMFRS problem. The method partitions the feasible region in a similar way as the branch-and-bound method; however, it only needs to keep a limited number of branches in each iteration. The method is also capable of incorporating efficient heuristics to converge faster and to rapidly reach efficient solutions. They compare the results of their method with the ones of static PMFRS algorithm and state that considerable performance benefits may be obtained by the use of flexible resources. They also report that the improvement increases with the number of machines.

Finally, Li, Wang & Lim (2003) propose a genetic algorithm-based approach for the $PD|res1\cdot\cdot|C_{\max}$ problem. The proposed approach incorporates encoding scheme which gives priorities to jobs and two decoding schemes. The first decoding scheme provides a general performance with lower complexity while the second one provides better performance with higher complexity. They use the first decoding scheme on all chromosomes but use the second scheme on those chromosomes with

better fitness values to improve their quality further. They also propose three kinds of lower bounds to evaluate the proposed genetic algorithm approach. Their computational results show that the average percent gap of the proposed algorithm is generally within 5%.

As already declared, CP, from the field of artificial intelligence, is an alternative solution technique to classical IP methods particularly in scheduling, sequencing and strict feasibility problems. For a related problem to RCPMSPs (see Hooker, 2005, 2006; Chu & Xia, 2005), CP has been utilized as a part of the solution procedure. Since the amount of resources for each facility is predetermined, these problems may be decomposed into a series of single-facility sub-problems by a master assignment model. The master assignment problem may be solved by an IP model, and smaller-size scheduling sub-problems may easily be solved by suitable CP models. In case of infeasibility of CP scheduling sub-problems, a valid cut (no-good cuts or cuts based on dual information of the sub-problems) may be generated and added to the master problem. The procedure goes on until all sub-problems become feasible. Hooker (2005, 2006) and Chu & Xia (2005) utilize such a technique for small and medium size scheduling problems. These papers show that the proposed decomposition algorithms reduce the solving time in comparison to the IP model. However, as already stated, the absence of additional resource(s) other than the main resource (machine) takes us away from classifying these problems into RCPMSPs.

3.6 Other Important Issues

Other than the significant characteristics of the surveyed papers discussed above, one important point is that, almost none of the studies in the literature deals with real life problems, although most of the related problems are encountered in real life environments with their own characteristics and necessitate industrial-size data to be solved. Moreover, among the studies with computational experiments, the majority of papers excluding a few ones (Blazewicz, Kubiak & Martello, 1993; Li, Wang & Lim, 2003; Ventura & Kim, 2003) deal with small or medium size problems. Blazewicz, Kubiak & Martello (1993) consider problems up to 1000 jobs and up to

10 additional resource types, with arbitrary resource sizes and resource requirements but with only two parallel machines. Ventura & Kim (2003) deal with up to 300 jobs, three additional resource types and five machines, however assumes that all the processing times are equal. Li, Wang & Lim (2003) consider problems up to 400 jobs and 40 machines on a dedicated machine environment.

Since most of the RCPMSPs studied are NP-hard (Blazewicz, Lenstra & Rinnooy Kan, 1983), it is unlikely to evaluate the results of proposed algorithms with optimal ones. Therefore, a number of studies (Edis, Araz & Ozkarahan, 2008; Li, Wang & Lim, 2003; Ruiz-Torres & Centeno, 2007; Ventura & Kim, 2003;) provide methods to obtain lower bounds. Lagrangian relaxation with a subgradient optimization procedure is an efficient technique to get tight lower bounds (Edis, Araz & Ozkarahan, 2008; Ventura & Kim, 2003).

Another significant point is whether to choose continuous time or discrete time IP formulations of the corresponding RCPMSP. In fact, both modeling approaches have their own strengths. Almost all continuous time formulations require the using of big-M constraints which results in weak LP-relaxation gap (see e.g., Pinto & Grossmann, 1997). On the other hand, discrete time formulations contain enormous number of variables (and also constraints) since the time indices should go into the scheme as an index of decision variables. However, discrete time formulations have two main advantages. Firstly, this type of formulations is easy to form in comparison to continuous time ones. Secondly, discrete time formulations generally perform better than continuous ones due to its efficient LP relaxation (Van den Akker, Hurkens & Savelsbergh, 2000). All the mathematical programming models given for the problems related to RCPMSPs in the literature are based on discrete-time formulations (see Daniels, Hoopes & Mazzola, 1996; Olafsson & Shi, 2000; Ventura & Kim, 2003)

There exist some other problem classes related to RCPMSPs in the literature. As Blazewicz, Lenstra & Rinnooy Kan (1983) state, the resource constrained scheduling with unit processing times is equivalent to a variant of the bin packing problem in

which number of items to be packed for each bin is restricted to *m* (number of machines). Therefore, the solution approaches for bin packing problems can be applied to this class of RCPMSPs. More information on the relationship between bin packing and resource constrained problems can be found in Garey, Graham & Johnson (1976) and Blazewicz & Ecker (1983).

In addition, one may wonder if there is a relation between RCPMSPs and resource constrained project scheduling problem (RCPSP). In both problems, schedules are constructed subject to resource constraints. However, RCPMSP differs from RCPSP in several aspects. RCPMSP requires an additional job-machine assignment sub-problem whereas the RCPSP does not incorporate machines. Furthermore, RCPSPs include precedence constraints in nature, while most of RCPMSPs do not contain precedence relations.

Scheduling issues occurring in batch chemical plants is also a related research field although the related studies of RCPMSPs do not cite any of the papers in this field. Indeed, scheduling of batch chemical plants are evaluated within their own isolated research field. Resource constraints including renewable resources also appear in the investigated problems of batch chemical plants (e.g., Lim & Karimi, 2003; Mendez & Cerda, 2004; Pinto & Grossmann, 1997). Readers are referred to Mendez, Cerda, Grossmann, Harjunkoski, & Fahl (2006) for a comprehensive review of this field.

3.7 Limitations of the Existing Literature and Distinguishing Properties of the Proposed Research

In this chapter overview of the studies related to RCPMSPs has been given. After reviewing the related literature, findings can be summarized as follows:

- In terms of machine environments, most studies deal with dedicated or identical machines. Uniform and unrelated machine environment are rarely studied. Moreover, to the best of our knowledge, no study so far considers the case of

machine eligibility restrictions and accordingly its effect on the schedule and performance criteria.

- A significant number of papers incorporate the case of resource-dependent processing times which additionally appends a resource allocation sub-problem into the scheme. The studies in this area, however, introduce some simplifying assumptions (e.g., machines are dedicated, allocation of resources to machines are static) to ease the problem at hand. More practical cases, the study of dynamic case, where the flexible resource can switch between machines during the schedule as well as the problems with unspecified job-machine assignment (i.e., the machines are not dedicated) are rarely studied.

- Most of the studies consider a single additional resource type. Moreover, the studies that handle more than one additional resource type consider unit (equal) processing times and/or two-three machines and/or unit size of additional resources with only 0/1 resource requirements. Thus, other versions with more than one additional resource type still remain a potential research area.

- Excluding a few papers, all the studies aim to minimize makespan. Other performance criteria may also be taken into consideration in the future studies. Moreover, all the studies deal with a single objective function due to the complex nature of this class of problems. The problems incorporating more than one criterion may further be considered in this research area.

- In terms of solution approaches, a number of studies give polynomial-time algorithms for some special cases of the problems. These papers generally handle simple cases with two or three machines, 0/1 resource requirements, or dedicated machine environments. However, many problems, yet, do not have a polynomial time algorithm. A significant number of studies are devoted to prove the NP-hardness of the related problems.

- Realizing that many practical problems are NP-hard, researchers focus on exact and approximation algorithms to solve the problems. Exact algorithms (e.g., B&B approaches, dynamic programming) are relatively few due to the combinatorial nature of the problem. Approximation algorithms, on the other hand, can be classified in three sub-groups: problem-based heuristic algorithms, ρ-approximation algorithms, and meta-heuristics. A significant number of studies focus on ρ-approximation algorithms with increasingly better worst case ratios, while problem-based heuristics and meta-heuristic approaches are relatively few. Furthermore, a significant number of studies relevant to ρ-approximation algorithms utilize relaxed mathematical formulations to solve the sub-problems of the entire problem at hand.

- Most of PMS problems with additional resource are encountered in real life environments with their own characteristics and necessitate industrial-size data to be solved. However, almost none of the papers surveyed deal with real life problems. Moreover, among the studies with computational experiments, the majority of papers deal with small or medium size problems.

In the light of the discussion above, distinguishing characteristics of the proposed research can be listed as follows:

- All research problems in this dissertation incorporate the case of machine eligibility restrictions and its effect in terms of flexibility measures on the schedules.

- Since each job can be processed on one of the machines from its eligible machine set, a more realistic case, i.e., the unspecified job-machine assignment, is taken into consideration in research problems.

- A more practical case, i.e., dynamic case where the flexible resource can switch between machines, is studied in all research problems.

- As the performance criteria, total flow time as well as makespan is considered in the proposed research.

- Lagrangian Relaxation is an efficient technique used in constrained optimization problems for obtaining tight lower bounds as well as generating efficient heuristic algorithms based on infeasible solutions of Lagrangian relaxation problem. As far as we know, only Ventura & Kim (2003) applied a Lagrangian-based solution procedure to a RCPMSP with identical machines. For the first problem case in this dissertation, a Lagrangian-based solution procedure is developed to obtain efficient results.

- It has already been stated that meta-heuristic approaches are rarely used in solving the investigated class of problems. Moreover, no studies so far, utilize CP in the solution procedures. In this dissertation, the solution procedures proposed for second and third problems utilize the strengths of CP in finding quick and efficient solutions for especially scheduling/sequencing part of the investigated research problems.

- Finally, the research problems in this dissertation are motivated by a real-world scheduling problem encountered in the injection molding department of an electrical appliance company. Moreover, the third problem case, i.e., real case study, is established on a series of problems with real data taken from the real environment.

3.8 Chapter Summary

This chapter gives a review and discussion of studies related to RCPMSPs in five topics. First, the papers have been discussed in terms of machine environment characteristics. Second, additional resource characteristics of the problems studied in the literature have been investigated and according to these characteristics, six main problem types have been identified. The related problem formulations have also been presented. Third, surveyed papers have been evaluated in terms of performance criteria. Fourth, a comprehensive review on the solution procedures has been

presented with their main characteristics. Fifth, other important properties of surveyed papers have been summarized. In the final section of this chapter, strengths and weaknesses of the related literature have been represented and distinguishing characteristics of investigated research problems and solution procedures have been listed.

The following chapter describes the three problem cases investigated in this dissertation in detail.

CHAPTER FOUR
PROBLEM STATEMENT

This chapter describes the research problems investigated in this dissertation. As already stated, there are three research problems all of which consider PMS problems with additional resources and machine eligibility restrictions. Since the research problems are motivated by a real world scheduling problem encountered in the injection molding department of an electrical appliance company, Section 4.1 gives the details of this manufacturing environment. Section 4.2 presents the common assumptions of the research problems. Then, in Section 4.3, the three research problems are described within the classification scheme defined before by giving their different properties. The complexity of research problems are also discussed within this section. Section 4.4 presents an illustrative example to clarify the investigated problems further. Finally, the related studies in the injection molding plants are briefly reviewed in Section 4.5.

4.1 Problem Definition

Due to the development of material technology, various plastic parts are widely used in electrical appliances. A plastic part is generally manufactured by an injection molding machine where a die must be setup on that machine. In large electrical appliance companies, product variety and make-to-order production cause hundreds of part-die tasks. These tasks are processed on many injection-molding machines with different dies to supply the assembling shop.

The injection-molding department, whose problem motivates this research, is one of the three departments in a supplier plant of an electrical appliance company. It produces several plastic parts for shipment to final assembly plants.

In the plant, each part has a die associated with it and its manufacture is completed at a single machine. The same die may be used to fabricate different parts by introducing material composition changes. Since the dies are so expensive, only

one unit is available from each type of die in the plant. Therefore, no parts that share the same die can be processed at the same time. Injection molding machines also differ in characteristics, and only a specific set of machines can be compatible with each die. The compatibility factors are technical factors like weight, pressure, quality requirements, etc. The case that each die may not be compatible with each machine type constitutes *machine eligibility restrictions.*

In addition, manufacturing process may require operators in accordance with the die type. A fixed number of machine operators are available in the plant for monitoring the machines, unloading and inspecting the plastic parts and trimming extra material. These tasks may not require an operator's full attention at one machine as discussed earlier by Bourland & Carl (1994). Based on data collected on the plant, there are three types of processes:

[1] *Manually operated processes* – requires exactly one operator along the processing.
[2] *Semi-automated processes* – One operator can deal with two machines simultaneously.
[3] *Fully-automated processes* – No operator is required during the processing.

The sum of operator requirement of parts being produced at any time period cannot exceed the available number of operators. The molding plant's objective is then to generate the schedule subject to constraints on capacity, machine eligibility and operator availability.

The problem described above is considered as a RCPMSP with machine eligibility restrictions and two types of additional resources (i.e., dies and machine operators).

4.2 Assumptions

A simplifying assumption may be considered on the use of dies. Recall that, in the plant, there is only one die from each die type, and a die may be used to fabricate

different parts. The parts that share the same die are assumed to constitute a *job string*. This basic simplifying assumption reduces the number of additional resources as well as problem size. Henceforth, dies are no longer restricted resources and considered as the job strings (or jobs) that will be assigned to machines along the scheduling horizon. Figure 4.1 illustrates the construction of the job strings. Suppose that we are given five parts associated with two dies. Part 1 and Part 4 that share Die 1 constitute job (string) 1; whereas Part 2, 3 and 5 that requires Die 2 constitute job (string) 2.

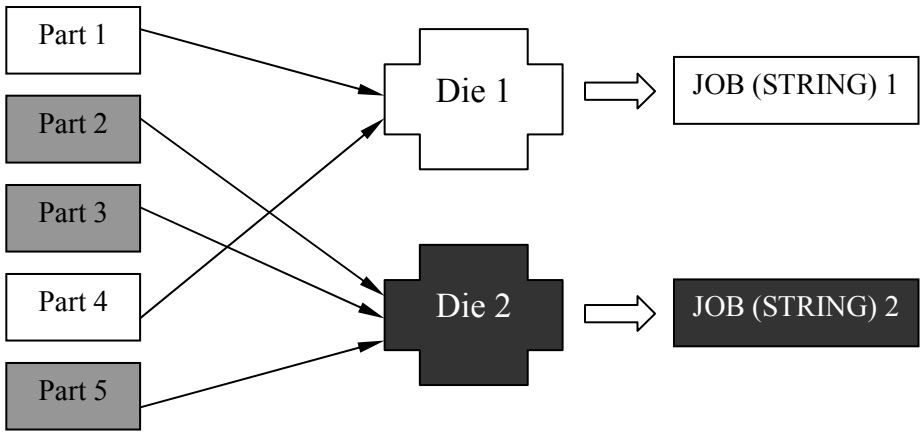


Figure 4.1 Construction of job strings

This research is also based on the following assumptions:

(1) All parts are available for processing at time zero.
(2) A machine can process at most one job string (or die) at a time.
(3) Machines and a fixed number of operator(s) are continuously available for the specified scheduling horizon.
(4) Preemption of job strings is not allowed, i.e., once they are started, they should be processed until completion.
(5) All processing times of job strings are deterministic.
(6) Setup time of dies is assumed to be one hour and incorporated in the processing time of job strings.
(7) Operator requirements of job strings are assumed to be fixed values, i.e. no operator, one-half of an operator, and one operator.

(8) All operators are fully cross-trained workers and operator transfer times between machines –in relation to job processing times- are negligible.
(9) Time window for scheduling of parts are fixed to the unit periods, i.e., one-hour length periods.

4.3 Research Problems

Motivating from the scheduling problem described above, we investigate three research problems in this dissertation. The first two problems are hypothetical, while the last one uses data from a real-life scheduling problem. All the research problems are classified in the field of RCPMSPs with machine eligibility restrictions. The aim is to develop exact and/or approximation algorithms to these problem cases each of which are NP-hard.

4.3.1 Problem Case I: $P|res1\cdot 2, M_i, p_i=1|\sum_i C_i$

A RCPMSP with one additional resource type with arbitrary size and up to two units of requirements with machine eligibility restrictions is considered as the first problem case. The number of jobs, number of machines and the size of additional resource are part of input. This version of RCPMSP is rather simpler than the other two problems since all job processing times are assumed to be the same. The objective is to minimize sum of completion times, i.e., $\sum_i C_i$.

For PMS problems with unit processing times, complexity of problems in terms of machine eligibility restrictions and additional resource constraints have already been discussed in Chapter 2 and Chapter 3, respectively. Since the presence of machine eligibility restrictions makes the problem a special case of unrelated machines, this problem with arbitrary number of machines and two units of the additional resource is most probably NP-hard.

4.3.2 Problem Case II: $P \mid res1 \cdot 2, M_i, p_i \mid C_{\max}$

The number of jobs, number of machines and the size of additional resource are also part of input in this problem case. Different from the first problem case, this version allows arbitrary processing times. Furthermore, the objective is to minimize makespan, $C_{\max}$.

It is well-known that ordinary PMS problem with m identical machines and makespan objective is NP-hard (Garey & Johnson, 1979). Moreover, Daniels, Hua & Webster (1999) state that a RCPMSP with three machines and one additional resource type with two available units ($b = 2$) and $R_1(i) \in \{1, 2\}$ is NP-hard. They also state that the RCPMSP is NP-hard, even if there is only one job per machine. Therefore, our investigated RCPMSP with arbitrary number of machines, one additional resource type with arbitrary size and $R_1(i) \in \{0, 1, 2\}$ including machine eligibility restrictions is also NP-hard.

4.3.3 Problem Case III (Real Case Study): $P36 \mid res1 \cdot 2, M_i, p_i \mid C_{\max}$

This problem case considers the actual problem of injection molding department with real data. In the injection molding plant, there are exactly 374 dies used in manufacturing of more than 1000 plastic parts on 36 injection-molding machines. The plant operates 12 hr per shift, two shifts a day, and five days a week. An MRP system explodes the dependent demand to generate weekly production orders for all parts. The managers of the plant aim to finish the manufacture of the current order quantities as soon as possible in order to leave an amount of time and resource (i.e., machine and operator) capacity to the newly incoming orders that may occur along the current week. Henceforth, the objective function is chosen as the minimization of makespan, i.e., time required to complete all parts.

Consequently, the aim is to develop a scheduling system that will receive MRP orders at the start of each week with the aim of minimizing makespan.

Note that, this problem case is also NP-hard, since it is a version of the second problem case with real data.

4.4 An Illustrative Example

Let us try to clarify the investigated RCPMSPs with arbitrary processing times by a small example. Assume that, a problem with 10 jobs, three machines and two units of a single additional resource (b = 2) is given. The processing times and resource requirements of jobs are as follows:

p_i = [8, 5, 4, 3, 2, 6, 1, 4, 7, 3], and
$R_1(i)$ = [0, 1, 2, 2, 0, 1, 0, 1, 1, 1], where i = 1,..,10.

Finally, M_i denotes the eligible machine set of job i and is given as follows.

$$M_i = [\{1,2,3\},\{1\},\{2,3\},\{3\},\{1,2\},\{1,2,3\},\{2,3\},\{2\},\{2,3\},\{1,2,3\}]$$

That is the first job can be processed on machine 1, 2, or 3; second job can be processed on only machine 1 and so on.

The objective is to minimize schedule makespan. Figure 4.2 through Figure 4.5 illustrate four different schedules. In all figures, each job is illustrated with its index and resource requirement in parentheses. For instance, 5(0) indicates the fifth job with no resource requirement. Similarly, 3(2) defines third job requiring two units of additional resource.

Figure 4.2 and Figure 4.3 illustrate infeasible schedules due to violating resource constraints and machine eligibility restrictions, respectively. Notice that, in Figure 4.2, the total resource requirement of jobs in time interval [2, 5] exceeds the available two units of the additional resource. In Figure 4.3, on the other hand, the third job is assigned to M1 which is not an element of its eligible machine set.

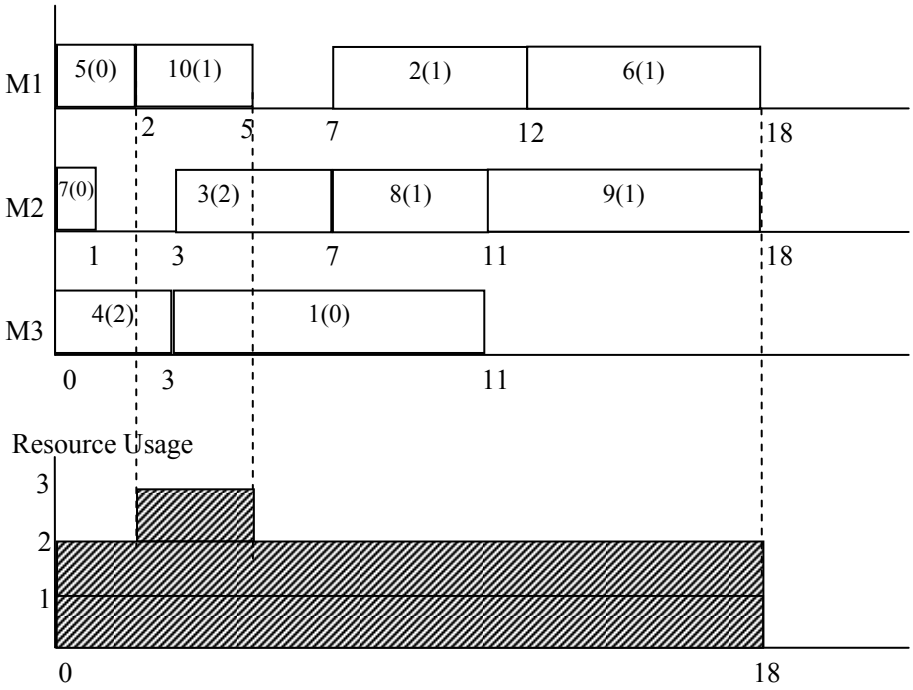


Figure 4.2 An infeasible schedule (violating resource constraints)

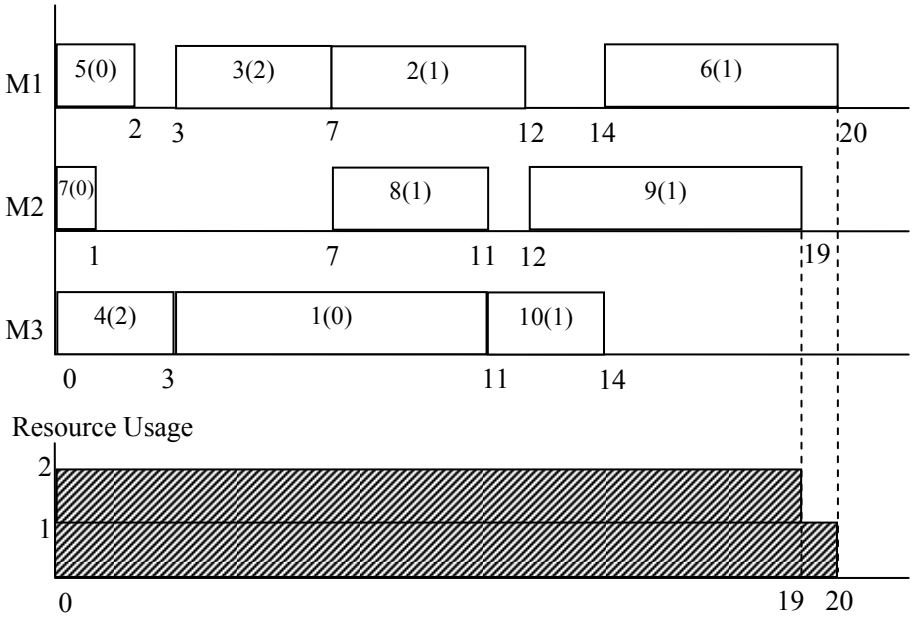


Figure 4.3 An infeasible schedule (violating machine eligibility constraints)

Figure 4.4 presents a feasible schedule which satisfies both the additional resource constraints and machine eligibility restrictions. Finally, an optimal schedule is illustrated in Figure 4.5 with a resulting makespan value of 20 time units. Notice that, in the optimal schedule, some time intervals remain idle for all three machines. It

may be thought that makespan could be reduced by moving some other jobs to these inserted idle times. However, this may violate the resource constraints and/or the machine eligibility restrictions by causing the current schedule become infeasible.

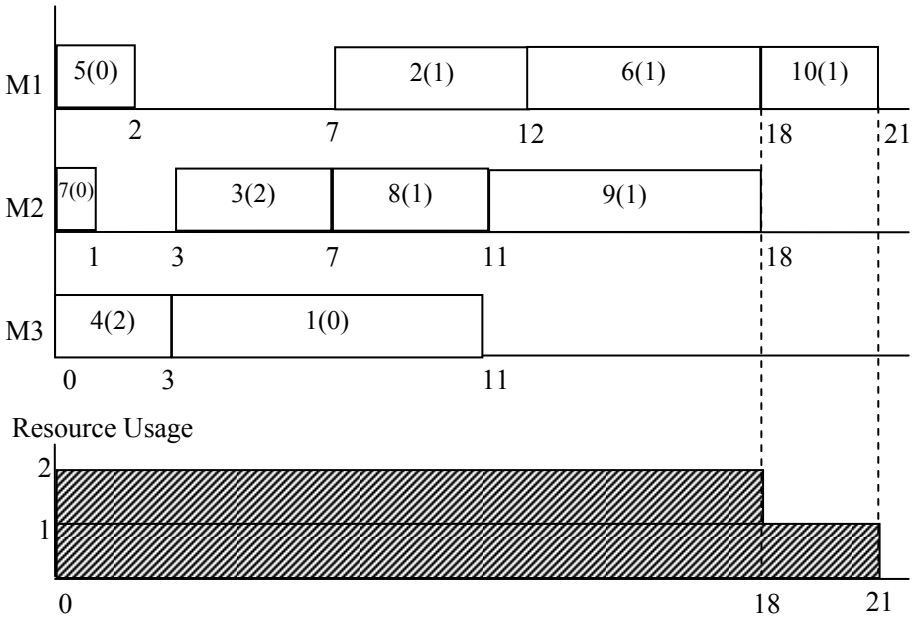


Figure 4.4 A feasible schedule

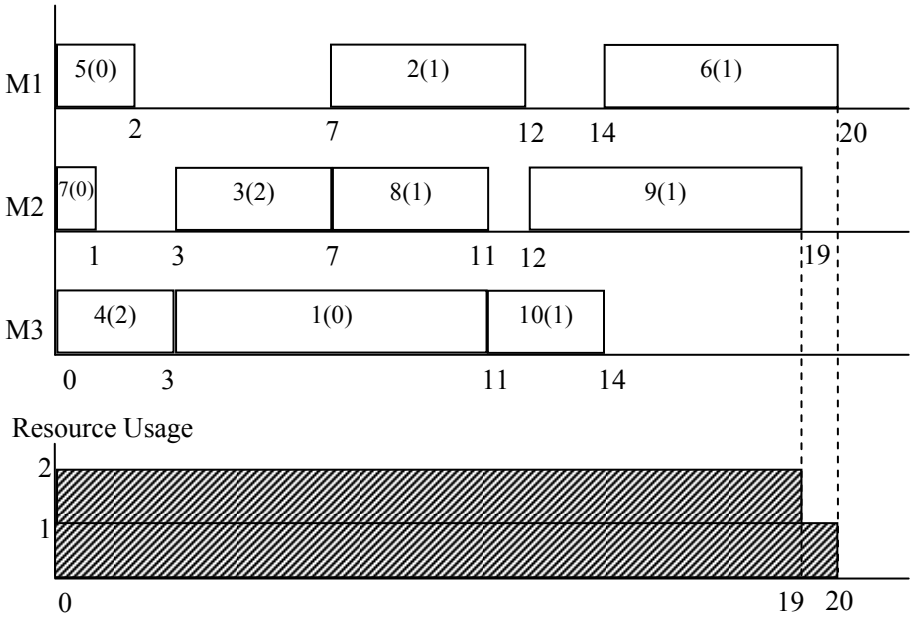


Figure 4.5 Optimal schedule

4.5 Related Research in the Injection Molding Plants

Since, in the injection molding plants, jobs may also require, besides machines, additional resources, (e.g., dies and machine operators) for their processing; the scheduling problem in these plants may fall into a special case of RCPMSPs with additional problem-based characteristics. There are a number of studies (e.g., Bourland & Carl, 1994; Chen, 2005; Chen & Wu, 2006; Gao et al., 1998; Lin, Wong & Yeung, 2002; Nagarur, Vrat & Duongsuwan, 1997; Nagendra, Das & Nathan, 2000; Tamaki et al., 1993) related to scheduling of injection molding machines in the literature. Since the problem is large-scale and difficult to solve optimally, solution approaches primarily focus on sequential based heuristics. All papers, except Bourland & Carl (1994), do not consider the limited operator availability. Bourland & Carl (1994) discussed a PMS problem with fractional operator requirements in a molding department of an automotive company. They modeled the problem hierarchically with a medium term production plan and a short-term schedule. They considered minimizing the setup and holding costs subject to limited number of operators. However, their problem does not include machine eligibility restrictions.

Different from these studies, all research problems investigated in this dissertation consider machine eligibility restrictions and common shared resources together.

4.6 Chapter Summary

This chapter has introduced the real scheduling environment which motivates the investigation of all three research problems. By giving the common assumptions to be considered, all three research problems have been explained with their distinct properties and complexity discussions. An illustrative example has also been presented to clarify the research problems in the mind of readers. Finally, a short review of studies related to scheduling efforts in injection molding plants has been given. The following chapter presents an overview of tools employed in solving the research problems.

CHAPTER FIVE
OVERVIEW OF THE SOLUTION TOOLS EMPLOYED IN THE PROPOSED RESEARCH

This chapter briefly explains the solution tools employed in the dissertation. Section 5.1 gives an overview of the well-known modeling technique Integer Programming (IP) with its main properties. Section 5.2 introduces Lagrangian Relaxation, Lagrangian heuristics and their applications in solving IP problems. Section 5.3 introduces Constraint Programming (CP) which is an alternative modeling technique for combinatorial optimization problems. This section gives main properties of CP and its applications in especially scheduling problems. A comparison between IP and CP techniques with their own strengths and weaknesses, as well as IP/CP integration schemes are also presented and discussed in this section.

5.1 Integer Programming

The roots of integer programming (IP) depend on linear programming (LP). LP is a subset of mathematical programming from the field of operations research (OR). OR is a discipline that deals with the optimization and control of systems. The term "programming" is used here as a synonym for optimization. (Eiselt & Sandblom, 2007, p.45)

Mathematical programming problems are mathematical models that attempt to model a real-life situation by using decision variables and parameters. Parameters are inputs given to the model whose values are known in a priori, while decision variables are numbers that will be determined in the solving process of the mathematical model.

All mathematical programming problems consist of two components; constraints and objectives. Constraints are established based on the restrictions on the problem definition. Typical examples are capacity constraints that limit resources (e.g., budget constraints, physical and/or chemical limits), and assignment constraints that

should be satisfied. It should be noted that constraints cannot be violated. On the other hand, objective(s) express the wishes of the decision maker. Most optimization models employ a single objective function.

In the light of above definitions, an LP model can be formulated in matrix notation as follows:

$$\begin{aligned} &\text{minimize} && cx \\ &\text{subject to} && Ax \geq b \\ & && Bx \leq d \\ & && x \geq 0 \end{aligned}$$

In this example, the first row represents the objective function which is to be minimized, the following two rows present the constraints and the last row give the domain of decision variables, $x = \{x_1, x_2, ..., x_n\}$. Notice that, in the above representation, the set of decision variables (x) is able to take all non-negative real values.

In the above formulation, if the set of x is restricted to take only discrete values i.e., $x \in Z^+$ the model is named as integer programming (IP). Moreover, if the set of x is restricted to take only binary values i.e., $x \in \{0,1\}$ the program is referred to as 0-1 IP. Finally, if some elements of x are allowed to take continuous values while the other ones are again restricted to take discrete values, the program is denoted as mixed IP (MIP).

It should be noted that, the constraints and objective function in IP must be linear. The strengths of IP/MIP include (Milano & Trick, 2004, p.15):

- Constraints are handled simultaneously through the linear relaxation, allowing arbitrary sets of linear constraints to be treated as a global constraint.
- Continuous variables can be handled naturally and efficiently along with the discrete variables.

- Bounds are generated to give deviation from optimality even when optimal solutions are not proven.

Mathematically, IP models require much more time to solve optimally in comparison to similar sized LP models. To solve LP models optimally, Dantzig (1963) developed a computationally efficient algorithm, named “simplex method”. On the other hand, although there are a wide variety of methods for solving IP, unlike LP with the simplex algorithm, there exists no universal algorithm with an efficient performance guarantee for solving IP models. The most successful algorithm so far found to solve IP problems is the branch and bound (B&B), which is heavily based on relaxations, i.e., easier sub-problems are obtained by removing or relaxing some constraints. Relaxations provide bounds on the objective function in order to prune suboptimal parts of the search tree. Another important aspect in solving IP problems is the use of cutting planes, i.e., linear inequalities which can be added to the relaxation to better approximate the geometrical structure of the original problem. Almost all commercial optimization packages offering IP/MIP solvers mainly use B&B algorithms enriched by some cutting plane methods. These two methods are briefly explained below:

Branch & Bound Methods. IP problem is first solved as an LP by relaxing the integrality constraints. If the resulting LP solution is integer, the problem is solved; otherwise a tree search is performed. Let us suppose that x is an integer variable whose optimum value x^* is fractional after solving the problem as an LP. Then, there are two alternatives to explore: $x \leq x^*$ or $x \geq x^*$. For each alternative, a copy of the problem is created, and one of the two constraints is added to it. This way of creating two sub-problems is called branching. Now, each sub-problem may be solved as an LP again. If the optimum solution of LP is feasible for the IP, this solution is recorded as the best one so far available. Otherwise, the sub-problem must further be partitioned into two sub-problems by again imposing the integer conditions on one of its integer variables that currently has a fractional optimal value. This process is repeated until a feasible solution is found or infeasibility is detected. Naturally, when a better integer solution is found for any sub-problem, it should replace the one at

hand. During the process, in any node, if the optimal solution of a relaxed sub-problem yields a worse objective value than the best available, there is no need to continue the search from that node of the tree since it cannot lead to a better solution. This process, named bounding, increases the efficiency of computations.

Cutting Planes Methods. This method is pioneered by Gomory (1963). As with B&B methods, first IP problem is solved as an LP by relaxing the integrality constraints. If the resulting LP solution is integer, this solution will also be the integer optimum. Otherwise, extra constraints (cutting planes) are systematically added to the problem, further constraining it. This process repeats until an integer solution is found or the problem is shown to be infeasible. (Williams, 1999, p.151-153)

Consequently, IP/MIP has the following characteristics (Milano & Trick, 2004; Thorsteinsson, 2001b):

- A complete approach.
- Optimization is done using objective function as a guide.
- Efficient for many classes of problems.
- Inflexible, e.g., restricted to linear constraints and 0-1 variables.
- All constraints are evaluated simultaneously.
- Little influence on search.

As discussed above, IP utilizes relaxation methods on the objective function. Beside the LP relaxation, an alternative way, Lagrangian relaxation, is also widely used to obtain tight lower bounds on the objective function and to generate near optimal solutions. The following section explains Lagrangian relaxation and its use for solving IP problems.

5.2 Lagrangian Relaxation and Lagrangian Based Solution Approaches for Integer Programming

Finding good solutions to combinatorial optimization problems requires the consideration of two issues (Beasley, 1995):

- calculation of a lower bound that is as close as possible to the optimum value.
- calculation of an upper bound that is as close as possible to the optimum value.

Figure 5.1 illustrates these two issues for a minimization problem. General techniques for deriving efficient upper bounds are essentially heuristic methods. These heuristics may be problem-specific or search-oriented algorithms. On the other hand, a common way to obtain the lower bound is relaxing the entire problem. Relaxation of an entire problem enlarges the set of feasible solutions. Assuming minimization, since there are more solutions to the relaxation than the original problem, the relaxation provides a lower bound on the optimal value.

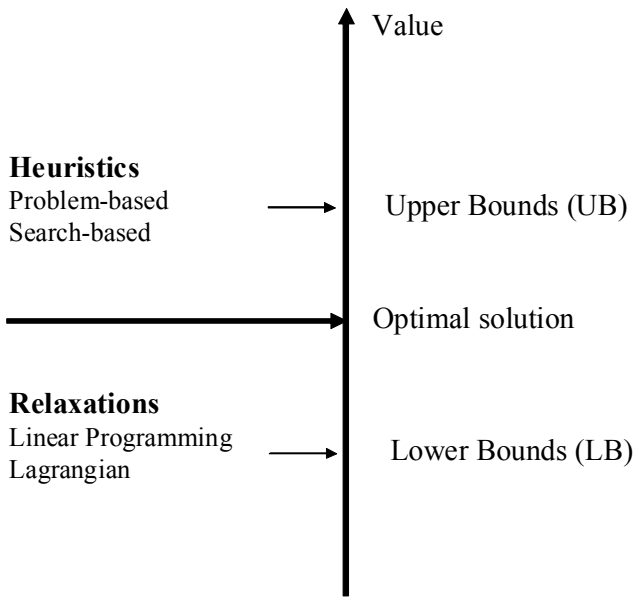


Figure 5.1 Upper and lower bounds in a minimization problem (Adapted from Reeves (1995, p.16))

One well-known relaxation technique is *Linear Programming (LP) relaxation*. In LP relaxation, an IP or Mixed IP (MIP) formulation of the problem is handled and the integrality requirements on the decision variables are relaxed. This results in an LP which is probably easily solved. The solution value obtained for this LP gives a lower bound on the optimal solution to the original problem (Beasley, 1995).

Another well-known (and widely used) technique which is available to find lower bounds is *Lagrangian relaxation*. Lagrangian method was developed by Held & Karp (1970, 1971) on the traveling salesman problem. Motivated by these studies, Fisher (1973) applied Lagrangian methods to scheduling problems. Geoffrion (1974) named this approach as "Lagrangian Relaxation". This technique mainly involves (Beasley, 1995):

1. taking an IP/MIP formulation of the problem.
2. attaching Lagrange multipliers to some of (complicating) constraints in this formulation and relaxing these constraints into the objective function.
3. solving (exactly) the resulting IP/MIP.

The solution value obtained from relaxed problem gives a lower bound on the optimal solution to the original problem. There are two basic reasons why this approach is well-known and widely used (Fisher, 1981; Beasley, 1995):

- Many combinatorial optimization problems consist of an easy problem (i.e., solvable by a polynomial algorithm) complicated by a relatively small set of side constraints. By transferring these side constraints into the objective function (step 2 above) we are left with an easy problem to solve and attention can be turned to choosing appropriate values for the Lagrange multipliers.

- Practical experience with Lagrangian relaxation has indicated that it gives very good lower bounds at reasonable computational cost.

Lagrangian Relaxation technique has been used with remarkable success in numerous applications to derive tight lower bounds and/or to construct good feasible solutions for difficult optimization problems (Frangioni, 2005). There exists a large body of literature on Lagrangian Relaxation, its extensions and applications for IP problems. A number of researchers (e.g., Beasley, 1995; Frangioni, 2005; Fisher, 1981; Geoffrion, 1974; Guignard, 2003; Lemarechal, 2001) also give comprehensive

reviews on Lagrangian relaxation methods for solving combinatorial optimization problems.

5.2.1 Lagrangian Relaxation

To clarify Lagrangian relaxation, let us consider the following general 0-1 IP problem (written in matrix notation) which is referred to as problem (P):

$$\begin{aligned} &\text{minimize} && cx \\ &\text{subject to} && Ax \geq b \\ & && Bx \geq d \\ & && x \in \{0,1\} \end{aligned} \tag{5.1}$$

Assume that, the constraints $Ax \geq b$ are *complicating* that the problem without these constraints would be much simpler to solve, whereas the constraints $Bx \geq d$ are relatively easy to handle. We define the Lagrangian relaxation problem (LRP) of (P) with respect to constraint set $Ax \geq b$ by introducing a Lagrange multiplier vector $\lambda \geq 0$ which is attached to this constraint set and brought into the objective function:

$$\begin{aligned} &\text{minimize} && cx + \lambda(b - Ax) \\ &\text{subject to} && Bx \geq d \\ & && x \in \{0,1\} \end{aligned} \tag{5.2}$$

In other words, the constraints $Ax \geq b$ have been dualized. Surely, the feasible solution set of LRP contains all feasible points of original problem P. The key point is that, for any $\lambda \geq 0$, the problem of LRP gives a lower bound on the optimal solution to the original problem (P).

The Lagrange multipliers (λ) in the above formulation determine the tightness of the lower bound. Therefore, it is desired to find λ values that give lower bound as close as possible to the optimal objective value (i.e., λ values that maximize the lower bound). This problem is denoted as Lagrangian dual problem (LDP):

$$\max_{\lambda \geq 0} \left\{ \begin{array}{ll} \text{minimize} & cx + \lambda(b - Ax) \\ \text{subject to} & Bx \geq d \\ & x \in \{0,1\} \end{array} \right\} \tag{5.3}$$

Ideally, the optimal value of LDP is equal to the optimal value of original problem. However, even with the best possible choice of the Lagrangian multipliers (λ), there is no guarantee that the penalty term in the objective function, i.e., $\lambda(b - Ax)$ will lead to a feasible solution. In fact, this is most often not the case: whenever this happens, the optimal solution of LDP is also optimal for (*P*) (Frangioni, 2005). If LDP and original problem (*P*) do not have equal optimal values, a "duality gap" (i.e., relative difference between two optimal values) arises.

5.2.2 Determination of Lagrange Multipliers

As discussed above, a significant point is to calculate the optimal values of Lagrange multipliers in order to maximize the lower bound. Two general techniques are available to optimize the λ values: *subgradient optimization* and *multiplier adjustment*.

Subgradient optimization method was proposed by Held & Karp (1971) and validated in Held, Wolfe & Crowder (1974). It is an iterative procedure which starts with an initial set of Lagrange multipliers and generates further multipliers in a systematic structure. Subgradient optimization is used to maximize the lower bound obtained from LRP. The matrix notation of relaxed constraints given in (5.1) can be written in the summation form as follows:

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i \text{ , } i = 1, \ \ldots, m \tag{5.4}$$

The basic steps of a general subgradient optimization procedure are presented below (Beasley, 1995):

1. Initialize the user-defined parameter π ($0 \leq \pi \leq 2$), initialize the upper bound (z_{UB}) by some heuristic. Set initial values for multipliers λ_i.
2. Solve the LRP with current set of λ_i to get a solution set x_j and the corresponding objective value, i.e., z_{LB} (lower bound).
3. Define *subgradients* G_i for the set of relaxed constraints evaluated at the current solution:

$$G_i = b_i - \sum_{j=1}^{n} a_{ij} x_j;\ i = 1, \ldots, m \tag{5.5}$$

4. Define the step size T by:

$$T = \pi(z_{UB} - z_{LB}) / \sum_{i=1}^{m} G_i^2 \tag{5.6}$$

   (Notice that, in this step, if $z_{UB} \cong z_{LB}$ the subgradient procedure is terminated.)
5. Update λ_i by

$$\lambda_i = \max\{0, \lambda_i + T \times G_i\};\ i = 1, \ldots, m \tag{5.7}$$

   and then go to Step 2 to resolve LRP with the new set of λ_i.

The subgradient optimization procedure in its above form would never terminate unless $z_{UB} \cong z_{LB}$. A usual way to terminate the procedure is as follows. First, an initial value is set to the user-defined parameter π. Through the subgradient iterations, the value of π is reduced subject to some rule. If π gets smaller than a pre-determined small value, then the subgradient procedure is terminated. A classical implementation is as follows (Beasley, 1995): If z_{LB} does not improve during (say) g consecutive subgradient iterations, the value of π is divided by two. Unless terminated in Step 4, the procedure is terminated if π gets a very small value (e.g., $\pi \leq 0.005$).

A significant point in the subgradient optimization procedure is that, in the problem formulation, all constraints which are to be relaxed have to be scaled since a large right hand side may dominate the step size expression for T (Beasley, 1995).

Practical convergence of subgradient procedure is unpredictable. In “good” cases, a saw-tooth pattern is usually observed in the earlier iterations, and consecutively, in the later iterations, a roughly monotonic improvement and asymptotic convergence to an optimal Lagrangian bound (i.e., the optimal objective value of LDP) is followed. In “bad” cases, on the other hand, the saw-tooth pattern always continues, or the Lagrangian bound even deteriorates (Guignard, 2003). Several studies (e.g., Fumero, 2001; Goffin, 1977; Lorena & Senne, 1999; Wang, 2003) focus on improvements of convergence rates of subgradient methods.

Beasley (1995) states that subgradient optimization has widely been used in the literature in conjunction with Lagrangian relaxation due to two main reasons:

- It provides good quality lower bounds for a wide variety of combinatorial optimization problems
- It can directly be applied to the relaxed constraints in many different problems.

Another procedure to determine the values of Lagrange multipliers is *multiplier adjustment*. This heuristic (Beasley, 1995):

- generates a starting set of Lagrange multipliers
- tries to improve them in some systematic way so as to generate an improved lower bound
- repeats the procedure if an improvement is made.

The multiplier adjustment procedure is easily applicable and usually produces an increase at each iteration; however, the final lower bound can be poor. Moreover, it requires special arrangements for different problems (Beasley, 1995). Therefore, it has received quite less attention in the literature in comparison to subgradient optimization procedure.

5.2.3 Lagrangian Heuristics

Since Lagrangian relaxation algorithms offer a lower bound (in minimization) to the original problem, it is often used as a measure of efficiency of the solution

obtained by a proposed heuristic. On the other hand, many researchers take the infeasible solution of LRP and try to convert it into a feasible solution for the original problem by suitable adjustments, i.e., by modifying the solution to correct its infeasibilities while keeping the deterioration of objective function small (Guignard, 2002). Such a solution procedure is named “Lagrangian heuristic” (LH), since it begins with the solution of LRP (Luh & Hoitomt, 1993).

Lagrangian heuristics are essentially problem dependent. Notice that, a LH should also satisfy constraints relaxed in LRP. The efficiency of the obtained schedule depends on the efficiency of the LH proposed. The resulting feasible solution constitutes an upper bound on the optimal solution. Obviously, a LH may not give an optimal solution in one trial. However, at each time of solving LRP through numerous subgradient iterations, LH takes an opportunity to transform the (possible) infeasible solution of various LRPs (with different infeasible solutions) to a feasible one for the original problem. Among the objective values corresponding to these feasible solutions, the best one is chosen. Therefore, it is common in practice to apply LH to the infeasible solutions of LRPs in each subgradient optimization procedure. This iterative process used to solve LRP, therefore, acts as a multi-start to classical heuristics (Frangioni, 2005). Surely, in such a structure, LH should give efficient solutions in a reasonable computation time.

The success of Lagrangian relaxation comes from clever implementations of methods for solving LDP, with powerful LHs applied at every subgradient iteration (Guignard, 2003).

Lagrangian relaxation procedure has, consequently, numerous advantages on solving combinatorial optimization problems:

- It provides so tight lower bounds that one may evaluate the performance of any heuristic according to this lower bound.
- It is not necessary to have special structures embedded in a problem in order to use Lagrangian schemes (Guignard, 2002).

- It provides good starting points for heuristic algorithms. Any feasible solution can be derived from the solution of LRP by a Lagrangian heuristic.
- Lagrangian bounds coupled with Lagrangian heuristics may help to prove the optimality if lower bound is (approximately) equal to the upper bound.

So far, the use of IP-based solution approaches for solving combinatorial optimization problems has been presented and discussed. The following section introduces CP which is an alternative technique of modeling and solving combinatorial optimization problems.

5.3 Constraint Programming and its Comparison/Integration with Integer Programming for Scheduling Problems

For a long time, IP has been the only technique for solving combinatorial optimization problems. Since the end of eighties, a different and declarative framework, named CP, from the fields of Artificial Intelligence and logics, has been developed (Milano & Trick, 2004). Since then, CP has been used as an alternative solution method to IP for solving combinatorial optimization problems. An overview of CP with its main techniques can be found in Smith (1995) and Brailsford, Potts & Smith (1999).

This section gives an overview of CP methods and its applications in combinatorial optimization problems giving further attention to scheduling problems. This section also presents a discussion on the comparison of IP and CP techniques as well as their simultaneous use particularly in scheduling problems. A part of this section can be found in Edis & Ozkarahan (2006).

5.3.1 Constraint Satisfaction Problem

CP is originally used to solve constraint satisfaction problems (CSPs). A CSP is defined by (Lustig & Puget, 2001):

- a set of variables,

- a corresponding set of domains that states the allowable values for each variable,
- a set of constraints over the variables which restrict the allowable combinations of variable values.

A solution to a CSP is an assignment of domain values to the variables that satisfies all constraints.

In CSPs, variables can assume different types including: Boolean (0 or 1), integer, symbolic, set elements and subsets of sets. Similarly, a variety of different constraint types are possible (Kanet, Ahire & Gorman, 2004):

- *mathematical*: completion time = start time + processing time
- *disjunctive*: job *i* and job *k* must be processed at different times
- *relational*: at most five jobs can be allocated at a machine
- *explicit*: only jobs *i*, *k*, and *v* can be processed on a machine.

In order to establish above constraint types, a CSP involves a wide variety of constraint operators such as: $=$, $<$, $\leq$, $>$, $\geq$, $\neq$, union, member, $\wedge$ (Boolean AND), $\vee$ (Boolean OR), $\Rightarrow$ (implies), $\Leftrightarrow$ (if and only if) (Kanet, Ahire & Gorman, 2004).

5.3.2 How to Solve a CSP?

Figure 5.2 presents a general algorithm for solving a CSP. It starts with defining variables and initializing their domains, and constraints to be stored in a constraint store. In block 2 of Figure 5.2, the domains of variables are reduced via constraint propagation and domain reduction algorithms (see Section 5.3.2.1). After this step, if a feasible solution is found, the algorithm ends (End1); otherwise, problem inconsistency is examined (block 4). If an inconsistency is not proven, then a search is performed using a search strategy for *branching* (block 4). Branching divides the main problem into a set of sub-problems by temporarily adding a constraint. Branching selects one of the branches and propagates all constraints again (block 2). If inconsistency is proven in block 4, which is referred to as a *failure*, the search tree is examined whether all sub-problems have been explored (block 5). If all branches

have been fathomed, the problem inconsistency is proved; otherwise the algorithm backtracks (block 7) and branches to a different sub-problem (block 6).

5.3.2.1 Domain Reduction and Constraint Propagation

Domain reduction is the direct application of a unary constraint c(x) to the variable x. For instance, if x is an integer variable with domain {0,1,…,10} and c(x) is x>6, then the domain of x becomes {7,8,9,10}. Constraint propagation is the propagation of changes in one variable's domain to the domains of other ones connected by constraints (Lustig & Puget, 2001). Constraint propagation uses constraints to make *arc consistency checking* between domains of the variables.

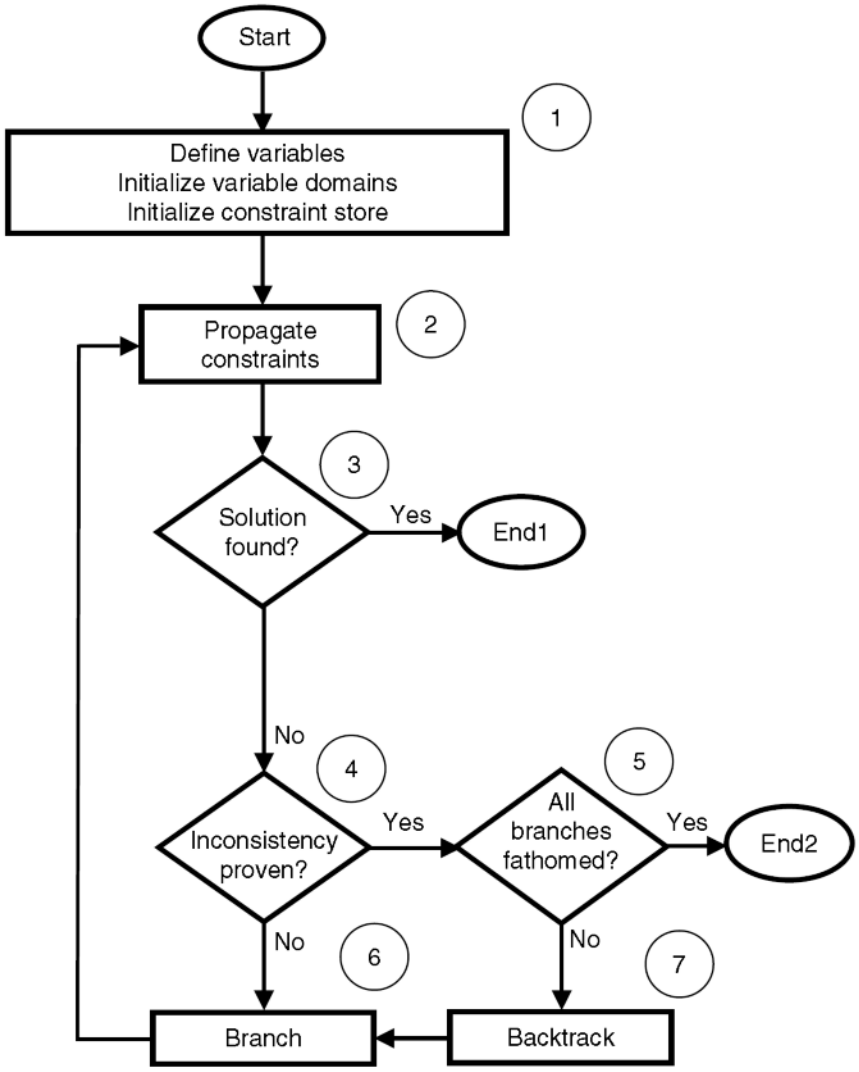


Figure 5.2 A general algorithm for solving CSPs (Kanet, Ahire & Gorman, 2004)

Figure 5.3 illustrates the concept of arc consistency with two variables, i.e., completion time of job 1, C_1, and start time of job 1, s_1. Assuming that processing time of job 1 is equal to 3, the constraint $C_1 = s_1 + 3$ should be satisfied. The domains of both variables are initially given as [0,1,2,3,4,5,6]. In part (b), the constraint $C_1 = s_1 + 3$ reduces the domain of C_1 using the domain values of s_1, making arc $C_1 \leftarrow s_1$ consistent. In part (c), the constraint now reduces the domain of s_1 using the updated domain values of C_1, making arc $C_1 \rightarrow s_1$ consistent (Kanet, Ahire & Gorman, 2004).

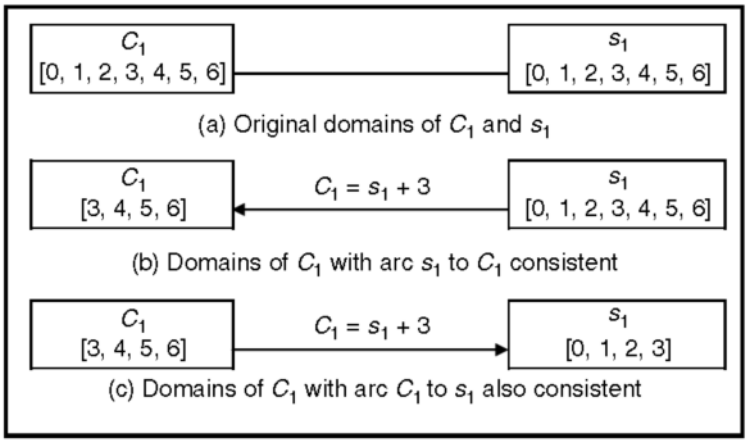


Figure 5.3 Arc consistency checking and domain reduction (Kanet, Ahire & Gorman, 2004)

5.3.2.2 Branching

The way of branching is also important in CP search tree. A significant feature of CP is its ability to define problem specific search procedures. By using an efficient search procedure, the search tree can be pruned in the former stages providing magnificent speedups in computation times (Smith, 1995). One often branches by first selecting a variable whose domain is not yet *bound* (reduced to a single value). Selection of the variable is often based on a *first-fail principle heuristic* such as "smallest current domain first". The selection process of variables is named "variable ordering". Once a variable is selected, then one chooses which value to assign the selected variable from its current domain. Again this choice can be made according

to a heuristic like “smallest value in the current domain first”. The process of selecting values is called “value ordering”.

More information on search procedures can be found in Van Hentenryck, Perron & Puget (2000).

5.3.3 Constraint Optimization Problem

A constraint optimization problem (COP) is defined as a CSP with an additional objective function. In COPs, as soon as a feasible solution is found, a new constraint is added into the constraint store, requiring further solutions to have a better value. Thus, CP solves a series of feasibility problems constrained to improve the value of objective function (Focacci, Lodi & Milano, 2002). This iterative process is repeated until inconsistency is found and all branches are fathomed. The last solution found is optimum.

It should be stated that pure solution algorithms designed for a COP do not utilize the strengths of linear relaxations and accordingly do not take the objective function as a guide.

5.3.4 The Richness of CP for Modeling and Solving Scheduling Problems

This sub-section gives the properties of CP that provides advantages on modeling and solving scheduling problems in comparison to IP methods. The attention is collected in three categories: *variable indexing*, *the strengths of constraint framework*, and *search algorithms* (Kanet, Ahire & Gorman, 2004).

5.3.4.1 Variable Indexing

Due to its roots on computer programming, CP has the ability of “variable indexing”, which allows one variable to be used as an index of other variable. This

capability provides a significant reduction in the number of decision variables required in the formulation.

For instance, assume a generalized job-machine assignment problem with $n \times m$ 0-1 decision variables, i.e., x_{ij} (i = 1, …, n, j=1, …,, m) where x_{ij} equals to one if job i is assigned to machine j, and zero otherwise. Although IP strictly requires such a variable definition, CP is able to handle this variable definition with only n decision variables (say $machine_i$) which take on the index of the machine to which a job is assigned. The equivalence relation between IP and CP decision variables can be given as follows:

$$x_{ij} = 1 \Leftrightarrow machine_i = j$$

5.3.4.2 The Strengths of Constraint Framework

As stated earlier, in contrast to IP, CP is not restricted to only linear constraints. CP allows using other types of constraints which are expressed below (Kanet, Ahir & Gorman, 2004):

Inequalities with Boolean Variables - Let us say a scheduling application incorporates two decision variables for two machines, x_1 and x_2 which take the value of 1 if a job is assigned and the value of zero otherwise. Further, it is desirable to assign a job to either one of two machines, not both. This situation is directly expressed in CP by the inequality $x_1 \neq x_2$. In IP, on the other hand, the same constraint can be specified as the equality $x_1 + x_2 = 1$ in an indirect way.

Logical Constraints - In many scheduling applications, overlapping of jobs are forbidden. Assume that in a two-job single machine scheduling problem, completion of job a must precede the start of job b, or, inversely, completion of job b must precede the start of job a, but not both. This situation is directly expressed in CP in the following form:

$$(a.\text{start} \geq b.\text{end}) \vee (b.\text{start} \geq a.\text{end})$$

On the other hand, in IP, this situation requires the use of Big-M constraints with an indicator variable:

$$S_a - C_b + \delta \text{ BigM } \geq 0$$
$$S_b - C_a + (1- \delta) \text{ BigM} \geq 0$$

where S_a and S_b denote the starting time of jobs, C_a and C_b present the completion times. The indicator variable $\delta = 1$ states that job *a* precedes job *b* and $\delta = 0$ implies job *b* precedes job *a*.

Global Constraints - These constraints apply across all or large subset of variables. For instance, in a job-machine assignment problem, it is desired to assign different jobs to each machine in an efficient way instead of using ≠ for all pairs of machines. The alldifferent( ) global constraint satisfies this situation in CP: alldifferent (machine). This constraint ensures that no two machines are assigned to the same job. It should be noted that such a global constraint has stronger propagation properties than a set of *n* inequality constraints (see Hooker, 2002). In IP, on the other hand, the alldifferent( ) concept can be implemented with a huge set of big-M constraints for every machine pair.

Notice that, the definitions of constraints in CP are compact and more declarative in representing real world situations. Moreover, IP may require much more constraints due to the requirements of using Big-M structures and indicator variables especially in formulating disjunctive constraints. On the contrary, CP may handle such structures by using logical expressions as well as special purpose global constraints.

5.3.4.3 Search

Since CP may be seen as an incomplete solution method, introducing an efficient search procedure is essential to obtain quick and efficient results. CP search methods rely less on particular mathematical structure of the objective function and constraints, but more on the domain knowledge of specific aspects of the problem.

Therefore, in contrast to IP, CP has the ability to define a search procedure concerning the variables of the program, to prune the branching tree in the earlier stages by giving priority to some variables.

Especially for scheduling problems, one may easily embed the scheduling knowledge into the design of the search procedure. In the scheduling literature, there has been a great knowledge of theorems and algorithms for specific scheduling problems. Similarly, there is a wealth of knowledge in scheduling about heuristic rules (e.g., LPT rule for minimizing makespan for PMS problems, LFJ rule for PMS problems with machine eligibility restrictions etc.) with empirical evidence or theoretical worst case bound analyses to show they provide good results. Such scheduling-specific domain knowledge is relatively easy to apply within the CP framework. Therefore, CP may be treated as a tool that complements scheduling algorithmic knowledge within a search procedure. (Kanet, Ahire & Gorman, 2004)

5.3.5 Comparison of IP and CP Methods for Scheduling Applications

So far, IP and CP have been explained with their main characteristics. The distinct characteristics of two methods are summarized in Table 5.1.

Table 5.1 Integer programming *vs*. Constraint programming (Thorsteinsson, 2001b, Focacci, Lodi & Milano, 2002; Milano & Trick, 2004)

Integer Programming (IP)	**Constraint Programming (CP)**
Optimization is done using objective function as a guide.	No relaxations; optimization is done without using objective function.
Inflexible, e.g., restricted to linear constraints and 0-1 variables.	Flexible, any mix of constraints (not only linear inequalities as in MIP).
All constraints are evaluated simultaneously.	Evaluates the effects of constraints sequentially.
Little influence on search.	Problem-specific search.

A number of researchers (e.g., Baptiste, Le Pape & Nuijten, 2001; Kanet, Ahire & Gorman, 2004; Baptiste, Laborie, Le Pape & Nuijten 2006) give comprehensive studies and/or reviews on the use of CP for solving scheduling problems.

Kanet, Ahire & Gorman (2004) state that one must consider the below issues while considering whether CP is an appropriate technique for solving the scheduling problem at hand:

- Since CP is more successful in reducing the domains of the variables which have finite domains, it is more appropriate for problems with integer (especially 0-1) decision variables
- CP is well suited for problems that tend to have a large number of logical, global and disjunctive constraints.
- CP operates well when there are a large number of interrelated constraints each of which includes only a few variables. Such a constraint framework provides better constraint propagation and domain reduction.

On the other hand, as compared to IP, the main weakness of CP when applied to a problem with an objective function (i.e., a COP) is that a lower bound may not exist (Lustig & Puget, 2001). Therefore, CP may often fail to prove optimality of the solutions due to absence of a lower bound.

So far, a number of papers (e.g., Heipcke, 1999) have dealt with a general comparison of IP and CP methods. More specifically, in the field of scheduling, a number of researchers use CP and compare it by IP in a wide range of scheduling problems. Smith et al. (1997) study the progressive party problem and state that only CP could give results with some modifications. Darbi-Dowman et al. (1997) consider a set of assignment problems and find that CP performs better. Darbi-Dowman & Little (1998) deal with four different combinatorial problems. Their results show that IP is efficient for problems with good relaxations while CP behaves well for highly-constrained problems but lacks a global perspective (i.e., the strengths of relaxations in IP). Mizrak Ozfirat, Edis & Ozkarahan (2006) compare IP and CP approaches for a course scheduling problem and find out that CP requires not only much less

number of variables in its compact formulation but also less computational time as compared to IP approach. Edis, Mizrak Ozfirat & Ozkarahan (2008) present a CP approach for a conference scheduling problem and obtain quick and efficient results. Ozkarahan, Edis & Mizrak Ozfirat (2009) compare IP and CP approaches for an operating room scheduling problem with sequence dependent preparation times between operations. Computational results show that IP provides relatively better performance, while CP gives very quick and practical solutions. Finally, in a recent paper, Kelbel and Hanzalek (in press) apply CP and MIP to scheduling problems with earliness-tardiness penalties. Numerical results show that CP performs better in all groups of test instances but fails to prove optimality in the given run time limit.

Consequently, above works infer that IP seems to be better for problems in which LP relaxations provide strong bounds for the objective function, while CP is better in sequencing, scheduling applications and strict feasibility problems. CP is also able to give quick and efficient results due to its compact and declarative framework, but lacks the global perspective and fails to prove optimality in many cases.

5.3.6 IP/CP Integration Schemes

Since IP and CP have their own strengths on solving complex combinatorial optimization problems as discussed above, combining these two methods has been an important research topic during the last years in various scheduling problems. A number of researchers (Focacci, Lodi, & Milano, 2002; Hooker, 2000; Milano, Ottosson, Refalo & Thornsteinsson, 2002; Milano, 2004; Milano & Wallace, 2006; Wallace, 2007; Hooker, 2007) present overview of integration schemes between IP and CP.

The most straightforward methods in integration schemes utilize the relaxation strength of IP methods in CP solvers. The strength of linear relaxation is related to its "global" view which is often omitted by CP. By combining all of the constraints of IP into a single linear program and inputting it to CP solvers as a global constraint (see Bosch & Trick, 2004); the linear relaxation is able to identify infeasibilities that

are not generated by any single constraint. This makes the linear relaxation a powerful tool for combining IP and CP. (Milano & Trick, 2004)

Such a straightforward combination of IP and CP is succeeded by a so-called "double modeling" as defined by Hooker (2000, 2002). This type of modeling requires a part of CP model constraints and a part of IP model constraints, or the constraints of both models as in many cases. These two distinct constraint sets are linked to pass domain reduction and feasibility/infeasibility information to each other. Such a double modeling approach provides utilizing linear relaxation of IP constraints to bound the possible values of uninstantiated variables. In doing so, the "global constraint" of the linear inequalities work together with logical constraints of CP to significantly improve the search. When keeping linear constraints as global constraints, it is important to keep only necessary ones that provide good bounds (Bosch & Trick, 2004). A number of optimization tools, e.g., Optimization Programming Language (OPL) Studio (ILOG, 2003), ECLIPSe (Wallace, Novello & Schimpf, 1997), provide frameworks for building combined IP/CP models by embedding the strengths of relaxations of IP to CP solvers. In literature, a number of researchers utilize double modeling in solving combinatorial problems. Rodosek, Wallace & Hajian (1999) combines MIP and CP, through CP Solver ECLIPSe and MIP Solver CPLEX, in which constraints can be handled by either one or both components. Jain & Grossmann (2001) present a combined IP/CP OPL model for an unrelated PMS problem and obtain more efficient results in comparison to IP and CP approaches alone. Topaloglu & Ozkarahan (2004) developed an IP/CP OPL model for single machine scheduling problem with sequence-dependent setups. Bosch & Trick (2004) apply a combined IP/CP OPL model to life design problems. Finally, Magatao (2005) developed a combined MIP/CP OPL model for scheduling operational activities in a multi-product pipeline and obtained efficient results.

On the other hand, a number of papers (e.g., Chu and Xia, 2005; Hooker, 2005, 2006; Jain & Grossmann, 2001; Thorsteinsson, 2001a) present integration schemes based on an IP/CP decomposition manner. This decomposition is performed with partitioning the original scheduling problem into a relaxed IP/MIP master problem and a series of CP scheduling sub-problems. The IP master problem includes the set

of constraints with good relaxations on the objective function, while the CP sub-problems cover the scheduling and sequencing constraints. As stated earlier, Hooker (2005, 2006) and Chu & Xia (2005) utilize CP technique based on an IP/CP decomposition manner for another class of PMS problems with resource constraints. Since these studies assume that additional resource constraints are related to parallel jobs in individual facilities (machines) rather than across all machines, it becomes straightforward to decompose the problem into independent single-machine sub-problems each of which can be handled individually by a CP model. These papers show that the proposed decomposition algorithms reduce the solving time compared with directly solving by MIP.

Consequently, IP and CP are two complementary approaches for solving complex combinatorial optimization problems. Combining these two methods has been an important research topic during the last years. So far, researchers have presented the individual strengths of IP and CP, and also have shown that the integration of both techniques may give better results than using two techniques separately. It has also been important for solving a number of real-world applications.

5.4 Chapter Summary

This chapter has presented the main properties of solution tools employed in the dissertation. First, an overview of IP technique has been given. Then, Lagrangian relaxation, Lagrangian heuristics and subgradient optimization procedures have been presented and discussed. Finally, overview of CP technique and its applications especially in scheduling problems have been given. Also a comparison and discussion of IP and CP techniques as well as their jointly use particularly in scheduling problems have been provided.

Once a background on the solution tools employed in the dissertation has been given, the following three sections present the proposed solution approaches for three investigated problem cases and accordingly discuss computational results.

CHAPTER SIX
LAGRANGIAN-BASED AND PROBLEM-BASED HEURISTIC APPROACHES FOR PROBLEM CASE I

6.1 Introduction

Machine scheduling problems are inherently difficult to solve via classical IP methods because of their combinatorial nature. When the additional resource constraints are included, scheduling problems become more complex. Since the use of exact methods is impractical for most real-world applications, it may be preferable to obtain rapid, easily applicable and near-optimal solutions. Hence, it is important to develop efficient heuristics for these problems.

This chapter deals with the first problem case i.e., $P\,|res1\cdot 2,\ M_i,\ p_i=1|\sum_i C_i$. Recall that, it is a PMS problem with one additional resource type, arbitrary resource size and up to two units of resource requirements. It also includes machine eligibility restrictions and assumes that all processing times are equal. The objective is to minimize total flow time. Differently from previous studies in the literature, the proposed research in this chapter considers machine eligibility restrictions and common shared resource cases together.

In this chapter, two heuristics are proposed for this problem case. The first one is a Lagrangian-based algorithm embedded into a subgradient optimization procedure, which solves a series of LRPs to maximize the lower bound and accordingly adjusts infeasible solutions of LRPs to obtain feasible schedules in order to minimize upper bound. The second one, on the other hand, is an independent problem-specific heuristic. The research in this chapter is based on Edis, Araz & Ozkarahan (2008).

The rest of this chapter is organized as follows. In Section 6.2, a 0-1 IP model for the considered problem is presented. Section 6.3 and Section 6.4 are devoted to explain proposed algorithms, i.e., a Lagrangian-based solution approach (LSA) with the subgradient optimization procedure and a problem-specific heuristic (PSH).

Computational results are given and discussed in Section 6.5. Finally, Section 6.6 summarizes the chapter and outlines the directions for the future research.

6.2 Problem Formulation

This section presents an IP model for the first investigated problem case i.e., $P\,|res1\cdot 2\,,\, M_i,\, p_i{=}1|\sum_i C_i$. Recall that, in addition to the assumptions presented in Section 4.2, all processing times of jobs are assumed to be equal in this problem case.

The following notation is used within the formulation of IP model.

Indices and Parameters

i: index of job strings (or dies) to be scheduled. $i = 1, ..., n$

j: index of parallel machines. $j = 1, ..., m$

t: index of time periods in the scheduling horizon. $t= 1, \ldots, T$

res_i: the amount of operator required by job i.

b: available number of operators for the scheduling horizon.

$$A_{ij} = \begin{cases} 1, & \text{if machine } j \text{ is compatible with job } i \\ 0, & \text{otherwise.} \end{cases}$$

Decision Variables

$$x_{ijt} = \begin{cases} 1, & \text{if job } i \text{ is processed on machine } j \text{ at time interval } [t-1, t) \\ 0, & \text{otherwise.} \end{cases}$$

Using this notation, the following IP model is developed:

$$\text{minimize} \quad \sum_{i=1}^{n} C_i$$

subject to:

$$\sum_{j=1}^{m} \sum_{t=1}^{T} t\, x_{ijt} = C_i \qquad i = 1, ..., n \tag{6.1}$$

$$\sum_{i=1}^{n} x_{ijt} \leq 1 \qquad j = 1, \ldots, m \; ; \; t = 1, \ldots, T \tag{6.2}$$

$$\sum_{j=1}^{m} x_{ijt} \leq 1 \qquad i = 1, \ldots, n \; ; \; t = 1, \ldots, T \tag{6.3}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} res_i \; x_{ijt} \leq b \qquad t = 1, \ldots, T \tag{6.4}$$

$$\sum_{j=1}^{m} \sum_{t=1}^{T} x_{ijt} = 1 \qquad i = 1, \ldots, n \tag{6.5}$$

$$\sum_{t=1}^{T} x_{ijt} \leq A_{ij} \qquad i = 1, \ldots, n \; ; j = 1, \ldots, m \tag{6.6}$$

$$x_{ijt} \in \{0,1\} \qquad i = 1, \ldots, n \; ; j = 1, \ldots, m \; ; \; t = 1, \ldots, T \tag{6.7}$$

The objective function minimizes total flow time working with Equations (6.1). Constraints (6.2) ensure that no more than one job can be assigned to any machine at any time interval. Constraints (6.3) guarantee that each job can be processed on only one machine at any time interval. Constraints (6.4) ensure that total number of operators assigned to jobs at any time interval is less than or equal to the available number of operators, b. Constraints (6.5) state that each job must be processed. Constraints (6.6) ensure that a job cannot be assigned to an incompatible machine. Finally (6.7) states that all x_{ijt} are 0-1 variables.

Clearly, this entire problem can also be tried to be solved by using some optimization software. However one can easily observe that such tools may not handle large problem sizes (e.g., 100 or more jobs) and may fail to find a solution (even feasible) in a reasonable time due to the complexity of the problem. This is one of the reasons why heuristic algorithms are developed for this problem.

The proposed solution procedures, i.e., LSA and PSH, are presented in the following sections.

6.3 Lagrangian-based Solution Approach (LSA)

Lagrangian relaxation is a mathematical programming technique used in constrained optimization. Since Lagrangian relaxation algorithms offer a lower bound (in minimization) to the original problem, it is often used as a measure of efficiency of the schedule obtained by a proposed heuristic. On the other hand, as stated earlier, Lagrangian relaxation often generates infeasible solutions. However, one can easily adjust these infeasible solutions to obtain feasible ones by using a Lagrangian heuristic (Luh & Hoitomt, 1993).

There exists a large body of literature applying Lagrangian relaxation procedures to different machine scheduling problems. A number of researchers have also tried to solve PMS problems by using Lagrangian relaxation algorithms. Luh, Hoitomt, Max, & Pattipati (1990) proposed a two-step optimization methodology for scheduling independent jobs with due dates on identical parallel machines. A Lagrangian relaxation technique is applied to mathematical formulation of the problem. They also applied a subgradient method to solve the dual problem. To obtain a feasible schedule, they used the resulting dual solution to form an ordered listing of jobs and then applied a greedy Lagrangian heuristic to assign jobs to the machines. Luh & Hoitomt (1993) formulated identical parallel machine problems and solved them based on Lagrangian relaxation techniques. They reported that Lagrangian relaxation based methods obtain near-optimal solutions in reasonable computation times. Martello, Soumis, & Toth (1997) proposed lower bounds based on Lagrangian relaxation for makespan minimization on unrelated parallel machines. Yu, Shih, Pfund, Carlyle, & Fowler (2002) studied the unrelated parallel machines with several performance measures such as makespan, mean flow time, utilization etc. They proposed Lagrangian heuristics to obtain feasible schedules. Kedad-Sidhoum, Solis, & Sourd (2008) used Lagrangian relaxation to obtain tight lower bounds for the earliness tardiness scheduling problem on parallel machines.

Although many researchers have studied the use of Lagrangian relaxation algorithms for PMS problems with the aims of both obtaining good lower bounds

and producing efficient heuristics based on Lagrangian solution, to the best of our knowledge, only one study (Ventura & Kim, 2003) utilizes this technique for a RCPMSP. Ventura & Kim (2003) studied an earliness-tardiness PMS problem with additional resource constraints and unit processing times. They formulated a 0-1 IP model and used Lagrangian relaxation approach to obtain tight lower bounds. They also proposed a Lagrangian heuristic to find near-optimal solutions. The study of Ventura & Kim (2003), which also motivates the research in this chapter, has reported the efficiency of Lagrangian relaxation algorithms. However, they deal with identical parallel machines. The proposed research in this sub-section, on the other hand, takes machine eligibility restrictions into consideration.

The following sub-sections give the details of the proposed LSA.

6.3.1 Lagrangian Relaxation of the Problem

Lagrangian relaxation is an approach to handle computationally hard IP problems. Specifically, some sets of difficult constraints are dualized to create a LRP which is easier to solve (Ventura & Kim, 2003). In our problem formulation, constraint set (6.4) ensures that, at each time period, cumulative usage of additional resource is within the available number of units, *b*. This constraint set, indeed, complicates the entire problem and removing it converts the problem to an ordinary assignment-type problem. Therefore, constraint set (6.4) is dualized so that the associated LRP has a straightforward structure and can easily be solved. By relaxing constraint set (6.4), the following LRP is obtained:

$$\text{(LRP) } Z_D(\lambda) = \text{ minimize } \sum_{i=1}^{n} C_i + \sum_{t=1}^{T} \lambda_t \left[\sum_{i=1}^{n} \sum_{j=1}^{m} res_i x_{ijt} - b \right] \tag{6.8}$$

subject to (6.1), (6.2), (6.3), (6.5), (6.6), and (6.7).

where $\lambda = [\lambda_t]$ is the set of nonnegative Lagrangian multipliers for constraint set (6.4).

Lagrangian multipliers ($\lambda = [\lambda_t]$) in the above formulation determine the tightness of the lower bound. Therefore, it is desired to find λ values that give lower bound as close as possible to the optimal objective value. The objective of the Lagrangian dual program (LDP) is then to find $\lambda = [\lambda_t]$ that makes the lower bound as large as possible, i.e.,

$$\begin{aligned} &\text{(LDP)} \quad \text{maximize} \quad Z_D(\lambda) \\ &\quad \text{subject to} \quad \lambda \geq 0 \end{aligned} \tag{6.9}$$

As stated in Section 5.2, to determine the optimal values of $\lambda = [\lambda_t]$, there are two alternative methods: subgradient optimization procedure and multiplier adjustments. In the proposed solution approach, a subgradient optimization procedure is applied to maximize the lower bound.

The subgradient optimization procedure, in general, requires an initial feasible solution. Therefore, first, an initial heuristic (IH) which constructs an initial feasible schedule is proposed. This IH provides an initial upper bound for the subgradient optimization procedure. Remember that, the objective of LDP is to find λ values that make the lower bound as large as possible. The standard subgradient optimization procedure aims to solve a series of LRPs, by updating λ values. On the other hand, LRP often generates infeasible schedules. To convert these infeasible schedules to feasible ones, a problem-based Lagrangian Heuristic (LH) is also developed. The resulting objective value of LH provides an upper bound for the original problem and is accordingly used to update the current upper bound. Subgradient optimization procedure, consequently, aims to match these lower and upper bounds updated through the consecutive iterations. Figure 6.1 illustrates the improvement of upper and lower bounds through the iterations of the subgradient optimization procedure of LSA.

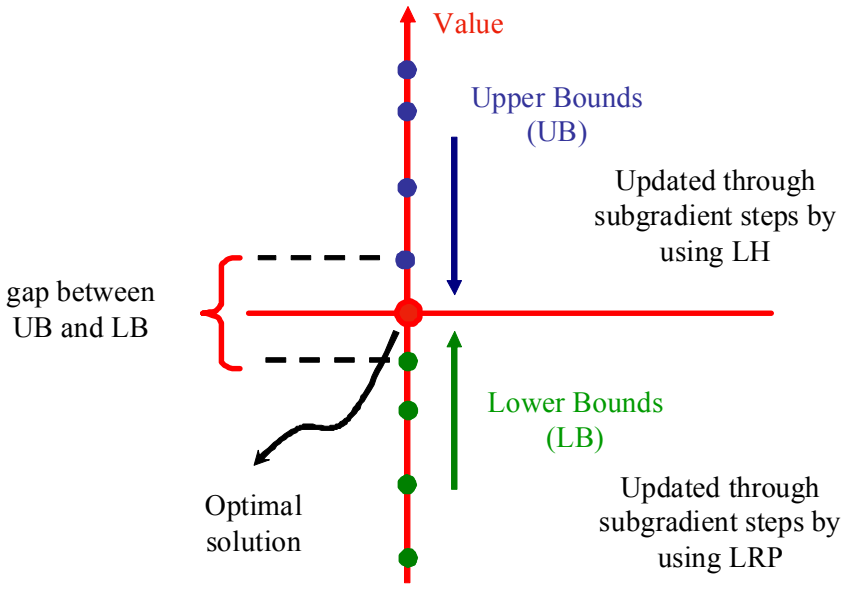


Figure 6.1 Improvements of upper and lower bounds through the iterations of the subgradient optimization procedure of LSA

The following sub-sections give the details of IH, LH and present the general structure of proposed LSA embedded into subgradient optimization procedure. Throughout these sub-sections, the following additional notation is used:

J	set of jobs
NRJ	set of non-resource jobs (i.e., the jobs that require no operator during its process)
$n(NRJ)$	the number of elements in set NRJ
$NRSJ_t$	set of non-resource jobs processed at time t
RJ	set of resource-jobs (i.e., the jobs that require a pre-determined amount of operator during its operation)
RSJ_t	set of resource jobs processed at time t
$n(RSJ_t)$	the number of elements in set RSJ_t
M	set of machines
M_i	eligible machine set of job i
R_t	the current usage of operators at time t
NM_i	the total number of eligible machines that job i can be processed on
LFJ	least flexible job first
MFJ	most flexible job first

UB^r updated upper bound at iteration r of subgradient optimization procedure

LB^r updated lower bound at iteration r of subgradient optimization procedure

6.3.2 Initial Heuristic (IH)

The aim of IH is to provide an initial upper bound (i.e., UB^0) for the subgradient optimization procedure. The basic idea of proposed IH is as follows. First, the jobs are arranged in decreasing order of their resource requirements and in case of ties, they are re-arranged by applying LFJ rule. Remember that, LFJ is a well-known dispatching rule for PMS problems with machine eligibility restrictions as discussed in Section 2.4. Then, beginning from the first time period, jobs in the arranged list are consecutively allocated to machines as early as possible subject to limitation on total number of available operators and machine eligibility restrictions. The details of IH algorithm are given below:

Step 1. Arrange the jobs in set J in decreasing order of res_i. If there are more than one job with same res_i, arrange them in non-decreasing order of NM_i, i.e., apply LFJ rule. It is assumed that, the jobs are indexed by their rank in the arranged list. Note that $[i]$ denotes the job with index i in the arranged list.

Step 2. Set $R_t = 0$, for all t; $x_{ijt} = 0$, for all i, j, t; set i=1, j=1, t=1.

Step 3. Assign job $[i]$ to machine j at time t if the following conditions are satisfied:

(i) $R_t + res_{[i]} \le b$ (ii) $j \in M_{[i]}$ (iii) $\sum_{l=1}^{n} x_{ljt} = 0$. Update $x_{[i]jt} = 1$, $R_t = R_t + res_{[i]}$, i=i+1 and go to Step 5. Otherwise, go to next step.

Step 4. If $j \ge m$, set t= t+1, j=1; otherwise, j=j+1. Go to Step 3.

Step 5. If i>n, set T = max $\{t \mid x_{ijt} = 1,\ i \in J, j \in M_i\}$, $z^0 = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t=1}^{T} t x_{ijt}$,STOP.

Otherwise, set j=1, t=1, go to Step 3.

Note that, feasible period length obtained by the solution of IH is also employed to determine length of scheduling horizon, T, to be used in LRP and PSH.

6.3.3 Lagrangian Heuristic (LH)

Since the constraint set (6.4) given in Section 6.2 is relaxed, at some time intervals, the cumulative usage of additional resource may exceed the available units of additional resource, *b*. In case of such a situation, the resulting schedule would be infeasible for the original problem. By a problem-based Lagrangian heuristic, we can detect infeasibilities and attempt to convert the schedule to a feasible one by suitable adjustments. The feasible solution obtained by such a heuristic constitutes an upper bound on the optimal solution. This upper bound is used to update UB^r through the iterations of subgradient procedure.

The proposed LH works in the following manner: At first, the current usage of operators at each time period, R_t and the resulting values of decision variables, x_{ijt}, are taken from current LRP. If the resulting schedule of LRP is infeasible, period *t* with the maximum R_t is picked. The set of resource jobs in period *t*, i.e., RSJ_t is then arranged by MFJ rule. Beginning from the first job in the arranged list, a suitable job is tried to be moved to an idle position before *t* subject to additional resource constraints and machine eligibility restrictions. If there is no feasible position for all jobs in the arranged list, among already assigned non–resource jobs within the current feasible period, a suitable job is searched to interchange with the first resource job in the arranged list. If again there is no suitable job that satisfies interchange conditions for any job in the arranged list, the first resource job is moved to the earliest feasible position after period *t*. In case of any job move or interchange of jobs, the corresponding R_t values are updated. Then, a new period with maximum current R_t value is picked and the same procedure is repeated until a feasible schedule is obtained. As a final step, in order to improve the objective function value further, non-resource jobs are tried to be moved into the earlier idle periods with respect to machine eligibility restrictions.

The detailed steps of the proposed LH are given below:

Step 0. Initialization. Take the resulting usage of additional resource at each time t (i.e., R_t) and x_{ijt} values from current LRP.

Step 1. If $R_t \le b$ for all t, go to Step 14. Otherwise, pick the period t with maximum R_t. If a tie occurs, pick the latest period among the alternatives as t.

Step 2. Arrange the jobs in set RSJ_t in decreasing order of NM_i. (i.e., apply the MFJ rule. It chooses the job that can be processed on the largest number of machines.) Break ties arbitrarily. It is assumed that, jobs are indexed by their rank in the arranged list of RSJ_t (Note that [i] denotes the job with index i in the arranged list). Set i=1.

Step 3. If $i \le n(RSJ_t)$, set t^*=1, j=1; go to next step. Otherwise, determine the current feasible period length, say t^c, and pick current machine of job [i], say $v \in M$. Set i=1, go to Step 7.

Step 4. If $t^* < t$, go to next step. Otherwise, $i=i+1$, go to Step 3.

Step 5. Assign job [i] to machine j at time t^* if following conditions are satisfied: (i) $R_{t^*} + res_{[i]} \le b$ (ii) $j \in M_{[i]}$ (iii) $\sum_{l=1}^{n} x_{ljt^*} = 0$. Update $x_{[i]jt} = 0$, $x_{[i]jt^*} = 1$, $R_t = R_t - res_{[i]}, R_{t^*} = R_{t^*} + res_{[i]}$, and go to Step 1. Otherwise, go to next step.

Step 6. If $j \ge m$, set j=1, $t^* = t^* + 1$, go to Step 4. Otherwise, $j = j+1$. Go to Step 5.

Step 7. If $i \le n(RSJ_t)$, set t^{**}=1, go to next step. Otherwise, set i=1, j=1, $t^{***} = t+1$, go to Step 12.

Step 8. Arrange the jobs in $NRSJ_{t^{**}}$ by applying MFJ rule. Break ties arbitrarily. Again, assume that the jobs are indexed by their rank in the arranged list. Set k = 1.

Step 9. If $t^{**} \le t_c$, go to next step. Otherwise, $i = i + 1$, go to Step 7.

Step 10. If $k \le n(NRSJ_{t^{**}})$, go to next step. Otherwise, $t^{**} = t^{**} + 1$, go to Step 8.

Step 11. Interchange job [i] on machine v at time t with job [k] on machine j at time t^{**} if following conditions are satisfied: (i) $j \in M_{[i]}$, $v \in M_{[k]}$ (ii) $R_{t^{**}} + res_{[i]} \le b$. Update $x_{[i]vt} = 0$, $x_{[i]jt^{**}} = 1$, $x_{[k]jt^{**}} = 0$, $x_{[k]vt} = 1$,

$R_t = R_t - res_{[i]}$, $R_{t^{**}} = R_{t^{**}} + res_{[i]}$ go to Step 1. Otherwise, $k = k$+1, go to Step 10.

Step 12. If $j > m$, set j=1, $t^{***} = t^{***}$+1; go to next step. Otherwise, go to next step.

Step 13. Assign job [i] to machine j at time t^{***} if following conditions are satisfied: (i) $R_{t^{***}} + res_{[i]} \le b$ (ii) $j \in M_{[i]}$ (iii) $\sum_{l=1}^{n} x_{ljt^{***}} = 0$. Update $x_{[i]jt} = 0$, $x_{[i]jt^{***}} = 1$, $R_t = R_t - res_{[i]}$, $R_{t^{***}} = R_{t^{***}} + res_{[i]}$, go to Step 1. Otherwise, j=j+1, go to Step 12.

Step 14. Arrange the jobs in *NRJ* by increasing order of their job number. It is again assumed that, the jobs are indexed by their rank in the arranged list. Set g=1.

Step 15. If $g \le n(NRJ)$, say the current period of job [g] is t' and set t''=1, go to next step. Otherwise, $z = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t=1}^{t_c} t\, x_{ijt}$, STOP.

Step 16. If $t'' < t'$, set j=1, go to next step. Otherwise, $g = g$+1, go to Step 15.

Step 17. If $j > m$, set $t'' = t''$ + 1, go to Step 16. Otherwise, go to next step.

Step 18. Assign job [g] to machine j at time t'' if following conditions are satisfied: (i) $j \in M_{[g]}$ (ii) $\sum_{l=1}^{n} x_{ljt''} = 0$. Update $x_{[g]jt''} = 1$, $x_{[g]jt'} = 0$; g=g+1, go to Step 14. Otherwise, $j = j$+ 1, go to Step 17.

6.3.4 Subgradient Optimization Procedure

Subgradient optimization is used to both maximize the lower bound obtained from LRP and update the upper bound by applying LH. The flowchart of the subgradient optimization algorithm is presented in Figure 6.2. A similar notation of Ventura & Kim (2003) is used within this procedure.

Beginning with the initial upper bound obtained by applying IH algorithm, the upper bound is consecutively improved by applying LH algorithm to the (possible) infeasible solutions of LRP.

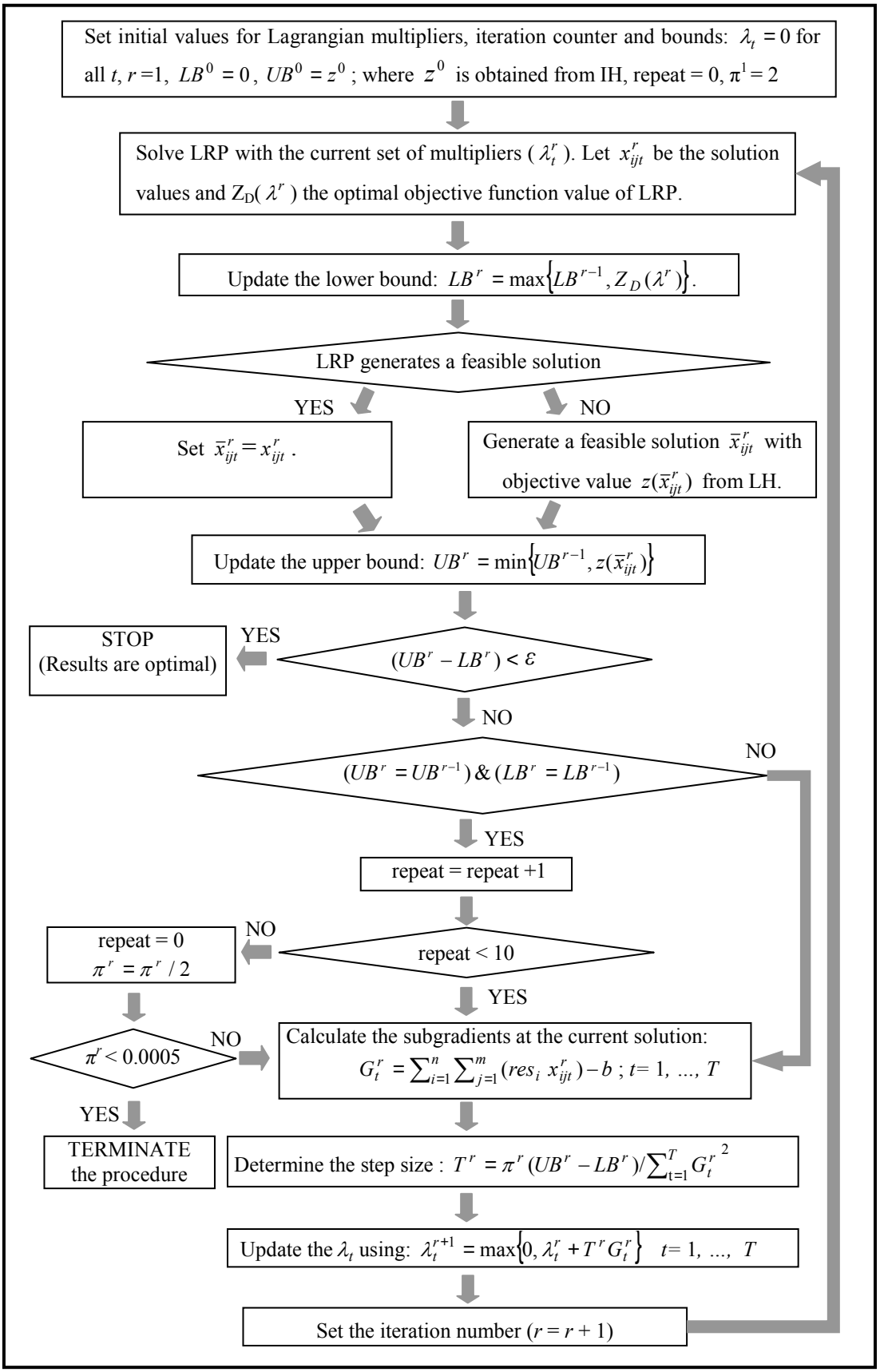


Figure 6.2 Subgradient optimization procedure

As stated earlier, subgradient procedure is terminated when the π parameter approaches to a very small value. After a preliminary computational test, it was found that the best convergence criterion is to set the initial value of π to 2 (i.e., $\pi^1 = 2$). If LB^r and UB^r do not improve during 10 subgradient iterations, the value of π^r is divided by two. If the gap (i.e., $UB^r - LB^r$) becomes less than pre-defined error ε (ε=1), the subgradient procedure is terminated. Since the value of objective function should be integer, $UB^r - LB^r < 1$ indicates that the proposed LSA converges to an optimal solution. If the procedure is not stopped with the error, it is terminated when π^r gets smaller than 0.0005.

Reconsider Figure 6.1, which illustrates the improvements of updated lower and upper bounds through the iterations of the subgradient optimization procedure. Assume that the optimal solution is somewhere on the vertical line. Through the subgradient iterations, lower bound is improved by the use of updated Lagrange multipliers (λ_t) within the corresponding LRP. Accordingly, upper bound is (possibly) improved by applying LH to the resulting infeasible schedule of distinct LRPs in each subgradient optimization procedure. Consequently, a small gap between upper and lower bounds is expected. Surely, the resulting upper bound is optimal if the absolute gap gets less than ε.

6.4 Problem Specific Heuristic (PSH)

PSH is an alternative solution approach to the considered problem and aims to find near-optimal solutions (i.e., upper bound). The basic idea of PSH is as follows. The jobs are separated into two sub-sets: set of non-resource jobs (*NRJ*) and set of resource-jobs (*RJ*). Beginning from the first time period, at first, the resource jobs are arranged in non-decreasing order of resource requirements and then allocated to machines with respect to limitations of the total number of available operators and eligible machine set of each job (M_i). Then, the non-resource jobs are arranged with LFJ rule and allocated to machines in a similar manner.

The details of PSH algorithm are provided below.

Step 1. Set $x_{ijt} = 0$ for all i, j, t; $R_t = 0$, for all t. Arrange the jobs in set *RJ* in non-decreasing order of res_i. If there are more than one job with same res_i, arrange them with LFJ rule. It is again assumed that, the jobs are indexed by their rank in the arranged list. Remember that [i] denotes the job with index i in the arranged list. Set i=1, j=1, t=1.

Step 2. Assign job [i] to machine j at time t if the following conditions are satisfied: (i) $R_t + res_{[i]} \le b$ (ii) $j \in M_{[i]}$. (iii) $\sum_{l=1}^{n} x_{ljt} = 0$. Update $x_{[i]jt} = 1$, $R_t = R_t + res_{[i]}$, i=i+1 and go to Step 4. Otherwise, go to next step.

Step 3. If $j \ge m$, $t = t + 1$, $j = 1$; otherwise, j=j+1. Go to Step 2.

Step 4. If $i > n(RJ)$, go to next step. Otherwise, j=1, t=1, go to Step 2.

Step 5. Arrange the jobs in *NRJ* with LFJ rule. It is again assumed that, the jobs are indexed by their rank in the arranged list. Set i=1, j=1, t=1.

Step 6. Assign job [i] to machine j at time t if the following conditions are satisfied: (i) $j \in M_{[i]}$ (ii) $\sum_{l=1}^{n} x_{ljt} = 0$. Update $x_{[i]jt} = 1$, i=i+1, go to Step 8. Otherwise, go to next step.

Step 7. If $j \ge m$, t= t+1, j=1; otherwise, j=j+1. Go to Step 6.

Step 8. If $i > n(NRJ)$, $z^{PSH} = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t=1}^{T} t\, x_{ijt}$, STOP. Otherwise, j=1, t=1, go to Step 6.

6.5 Computational Results

The LSA and PSH given in the previous sections are coded in OPLScript (ILOG, 2003). OPLScript is a script language that composes and controls OPL models. OPL Studio 3.7 (ILOG, 2003) is the development environment of OPLScript. All generated LRPs are implemented in OPL Studio 3.7 using a Pentium D 3.4 GHz 4 GB RAM computer.

In experimental studies, two levels of processing flexibility for the machines are determined using the process flexibility index (F_P) defined by Vairaktarakis & Cai (2003) which is discussed in Section 2.4. Remember that F_P gives a measure of

flexibility of machines in terms of machine eligibility restrictions. It takes values between 0 and 1, where $F_P = 0$ indicates that each job is able to be processed on only one machine and $F_P = 1$ describes a fully flexible case that each job is able to be processed on all machines. For all test instances, the cases that have F_P higher than 0.65 are defined as high processing flexibility cases, whereas the others that have a F_P less than 0.35 are considered as low processing flexibility cases.

The number of jobs is taken as 50 and 100. The number of machines is selected as three, five and eight with respect to number of jobs. The number of operators available is taken as one, two and three with respect to the number of machines. For each combination of parameters, 30 test problems were solved.

Remember that, three process types are defined in Chapter 4. Firstly, a job may require exactly one operator along its processing (i.e., resource requirement of the job is one). Secondly, one operator can deal with two machines simultaneously (i.e., resource requirement of the job is one-half). And finally, no operator is required during the processing (i.e., resource requirement of the job is zero). Therefore, for all test problems, operator requirements of jobs are generated with equal probability for the values 0, 0.5 and 1. Note that, for the computational purposes, in solving LRPs, operator requirements have been made integer by multiplying them by two. The number of operators has also been accordingly increased.

The detailed results of LSA and PSH related to all 240 test problems are given in Appendix B. A summary of these results are presented in Table 6.1. To evaluate the performance of proposed algorithms, we gathered average number of iterations, CPU time, average and maximum gap percent and number of optimal solutions for LSA; and average and maximum gap percent values with number of optimal solutions for PSH. Note that CPU time of PSH is negligible.

It should also be highlighted in Table 6.1 that the LSA gives solutions within reasonable times for all cases.

Table 6.1 Computational results

n	*m*	*# of Oper.* (*b*)	*Proc. Flex.* (F_P)	*Lagrangian-based Solution Approach (LSA)*				*Problem Specific Heuristic (PSH)*	
				Avg. # of Iter.	*Avg. Time(s)*	*Max.% (UB-LB)/LB*	*Avg.% (UB-LB)/LB*	*Max.% (UB-LB)/LB*	*Avg.% (UB-LB)/LB*
50	3	1	Low	148.70	21.86	2.92	0.97 (9)	11.25	2.70 (4)
			High	167.87	33.77	4.61	1.91 (3)	4.57	1.00 (11)
	5	2	Low	88.40	9.33	1.00	0.26 (20)	11.11	2.12 (3)
			High	120.50	19.06	1.02	0.35 (16)	1.02	0.42 (13)
100	5	2	Low	167.93	95.97	1.09	0.63 (0)	3.69	1.03 (1)
			High	162.83	132.33	1.59	1.03 (1)	1.71	0.34 (6)
	8	3	Low	150.73	88.40	1.09	0.52 (4)	3.58	0.81 (1)
			High	144.00	121.68	1.00	0.47 (7)	1.10	0.27 (15)

In Table 6.1, the numbers in parentheses represent the number of times that the algorithms achieved optimal solutions. In fact, LSA is able to prove optimality within its subgradient procedure when the difference between updated upper bound and updated lower bound becomes less than ε=1. The results of PSH are evaluated by using the lower bounds obtained from the subgradient procedure of LSA. Similar to LSA, the solution of PSH is proved to be optimal if the gap between its objective value and the corresponding lower bound is less than ε=1.

Although the aim of proposed heuristics is to reach near-optimal solutions, LSA and PSH attained optimal solutions in 60 and 54 test problems, respectively. When both heuristics are considered together, we reached optimal solutions in 85 out of 240 test problems.

The computational results show that LSA gives good results based on average percent deviation which represents the mean gap percent between the lower bound and upper bound obtained by subgradient optimization procedure. It should also be noted that subgradient optimization procedure produces very tight lower bounds for almost all cases. The proposed LSA attained less than 1% deviation from the optimal solution in 169 problems out of 240 test problems. The average gap percent of LSA for all problems is 0.77. PSH also gives good results. It attained less than 1% deviation from the optimal solution in 157 out of 240 test problems. The average percent deviation of PSH for all problems is 1.08.

Figure 6.3 compares the LSA and PSH in terms of average gap percent. It shows that LSA gives relatively better results in case of low processing flexibility environments. It ensures less than %1 deviation in 80.00% of test problems with low flexibility, while reaches less than 1% deviation in 60.83% of test problems with high flexibility. On the contrary, PSH provides relatively better results in case of high processing flexibility. While the number of problems that PSH reached less than 1% deviation is 60 (50%) among 120 problems with low flexibility, it provides less than %1 deviation in 97 (80.33%) out of 120 problems with high process flexibility. These results show that LSA provides superior performance than PSH in low flexibility cases whereas PSH performs better when the process flexibility is high.

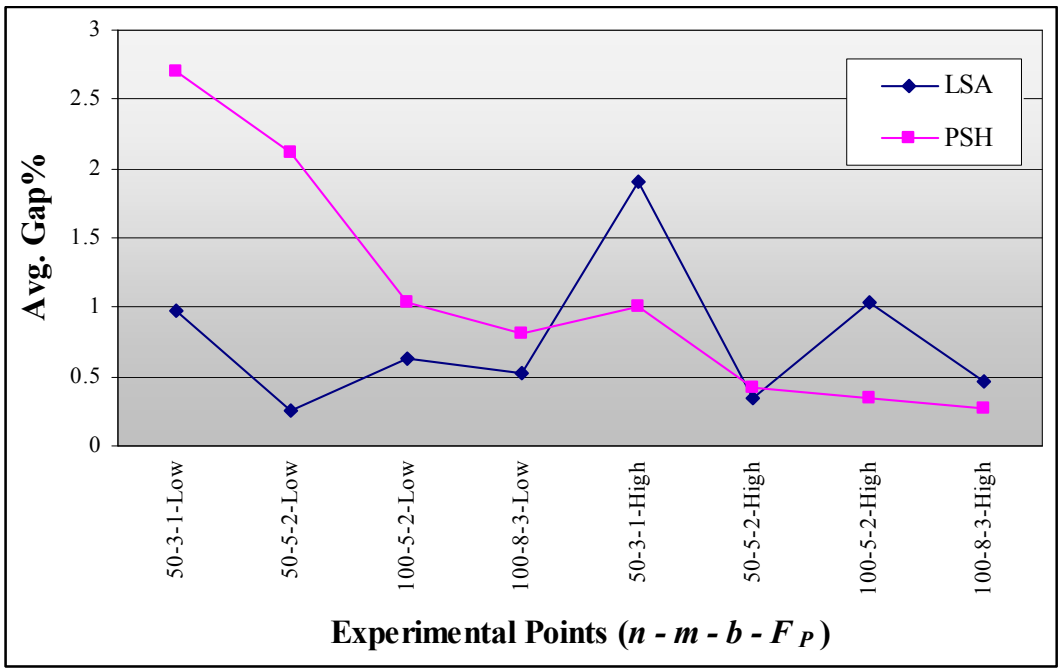


Figure 6.3 Comparison of average gap % in experimental points

Figure 6.4 compares the LSA and PSH in terms of maximum gap percent. As can be seen, the variance of PSH is very high especially in low processing flexibility cases. On the other hand, LSA produces more stable solutions in both low and high flexibility cases. The maximum gap percent values of LSA are significantly smaller than the ones of PSH in low processing flexibility cases, whereas it cannot be seen any remarkable difference in high flexibility cases.

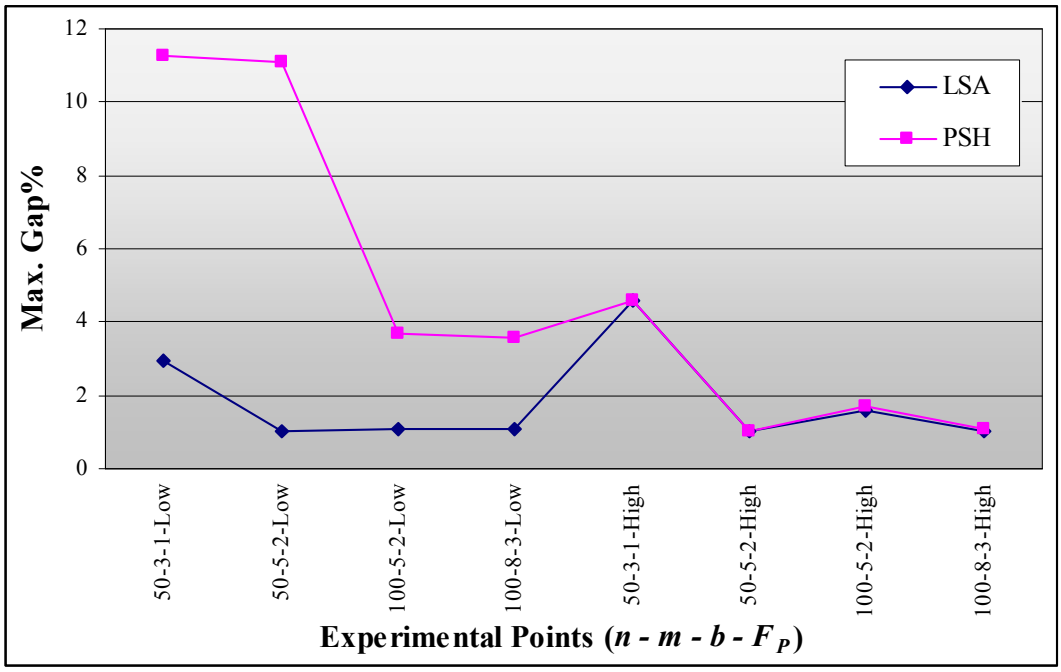


Figure 6.4 Comparison of max gap % in experimental points

A significant issue in Lagrangian based solution methods is the convergence of the upper and lower bounds. Figure 6.5 illustrates a convergence representation of a sample instance (i.e., n = 50, m =3, b = 1, F_P = Low-Sample No.1, see Appendix B).

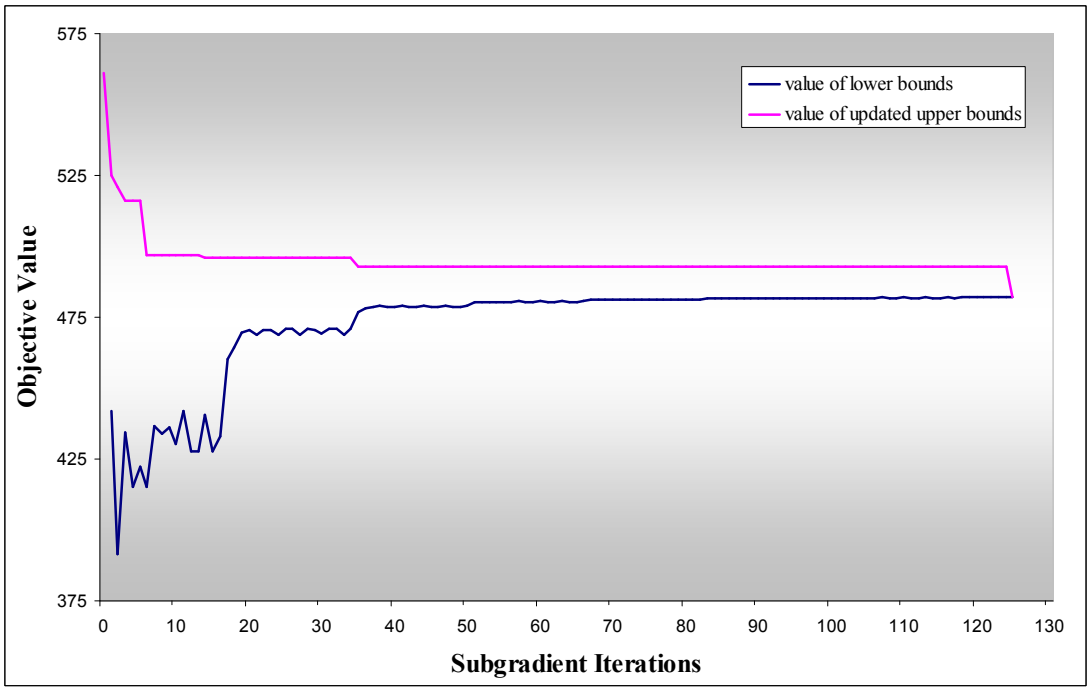


Figure 6.5 Convergence representation of a sample instance (50-3-1-Low-Sample 1)

As seen from Figure 6.5, the lower bound converges its approximate best value in about 70 iterations; however upper bound is able to converge the optimal solution in 125th iteration where the optimality of the resulting solution is also proved. Sample convergence graphics related to other seven sub-groups of test problems are presented in Appendix C. These figures show that the subgradient procedure provides tight lower bounds within a reasonable number of iterations and accordingly in a reasonable computation time.

6.6 Chapter Summary

This chapter has addressed the first research problem in this dissertation. At first, the 0-1 IP model of the problem is constructed. Then a Lagrangian-based solution approach (LSA) that involves an IH, a LH and a subgradient optimization procedure to obtain tight lower bounds and near optimal solutions is developed. Also, a problem specific heuristic (PSH) is independently proposed. By means of randomly generated instances of the problem, it is shown that the proposed algorithms produce not only very tight lower bounds but also efficient results with a small optimality gap. LSA gives superior results in low flexible machine environments, while PSH is relatively better in high flexible ones.

CHAPTER SEVEN
INTEGER PROGRAMMING (IP), CONSTRAINT PROGRAMMING (CP) AND IP-CP COMBINED APPROACHES FOR PROBLEM CASE II

7.1 Introduction

This chapter deals with the second problem case i.e., P $|res1\cdot 2, M_i, p_i| C_{\max}$. Recall that it is a PMS problem with one additional resource type, arbitrary resource size and up to two units of resource requirements. It also includes machine eligibility restrictions and allows arbitrary processing times. The objective is to minimize makespan. Different from previous studies in the literature, this problem case considers machine eligibility restrictions and common shared resource cases (i.e., machine operators shared by all machines) together.

In fact, a common way to represent a machine scheduling problem is to model it with IP/MIP. One of the methods to obtain the optimal solution of IP/MIP models is branch and bound algorithm (B&B). However, machine scheduling problems are inherently difficult to solve via classical B&B methods because of their combinatorial nature. Although it is possible to find optimal solutions via IP/MIP methods, it may require an enormous amount of computation time with the increasing problem size. When the additional resources are included, the problems become more difficult to solve via classical IP methods. A number of papers (e.g., Daniels, Hoopes & Mazzola, 1996, 1997; Daniels, Hua & Webster, 1999; Edis, Araz & Ozkarahan, 2008; Ventura & Kim, 2003) present IP/MIP models of different PMS problems with resource constraints. However, most of these papers use heuristic or meta-heuristic approaches to solve the problems.

As discussed earlier, in recent years, constraint programming (CP) has been used as an alternative solution method to the combinatorial optimization problems. CP is a suitable technique for sequencing and scheduling applications, strict feasibility problems and highly constrained problems (Darbi-Dowman & Little, 1998; Jain & Grossmann, 2001; Lustig & Puget, 2001; Smith et al., 1997). However, CP cannot

provide a global view on the problem since it does not utilize the advantage of relaxations.

Since IP and CP have their own strengths on solving complex combinatorial optimization problems, combining these two methods has been an important research topic during the last years for various scheduling problems (e.g., Chu & Xia, 2005; Hooker, 2005, 2006; Jain & Grossmann, 2001; Thorsteinsson, 2001a). It has also been important for solving a number of real-world applications, which cannot be solved by one of the approaches alone (Hooker, 2005). To the best of our knowledge, only Hooker (2005, 2006) and Chu & Xia (2005) utilize this technique based on an MIP/CP decomposition manner for a different class of scheduling problems with resources. These papers show that the proposed decomposition algorithms reduce the solving time in comparison to directly solving by MIP. Since these studies assume that additional resource constraints are related to parallel jobs in individual facilities (machines) rather than across all machines, it becomes straightforward to decompose the problem into independent single-machine sub-problems each of which can be handled individually by a CP model.

In this chapter, IP, CP and combined IP/CP models are developed for the second problem case. Since the resource constraints are across for jobs through all machines, our problem should be handled with a global model including all constraints. Therefore, rather than a decomposition approach as given by Hooker (2005, 2006) and Chu & Xia (2005), we propose a combined IP/CP OPL model by embedding an efficient search procedure into it. This IP/CP combined model is an instance of "double modelling" approach (Hooker, 2002) that is an exact combination of IP and CP models with linking variables and constraints. To the best of our knowledge, no study so far has proposed a combined IP/CP model for RCPMSPs with a common additional resource to minimize the makespan objective. Furthermore, the use of problem-based search procedures has not been adapted to the applications of CP and combined IP/CP approaches in the earlier RCPMSP studies. The proposed research presented in this chapter also analyzes the effect of machine eligibility restrictions in terms of process and balance flexibility measures for the investigated problem case.

The rest of this chapter is organized as follows. The proposed three models, IP, CP and IP/CP combined OPL models are presented in Section 7.2. Section 7.3 gives a discussion of CP-based search procedures and proposes two efficient problem-based search procedures including special characteristics of the investigated problem case. Section 7.4 gives implementation issues and computational results. Finally, Section 7.5 summarizes the proposed research in this chapter.

7.2 Proposed Models

The following sub-sections propose three optimization models for the considered RCPMSP: IP, CP and IP/CP combined OPL model.

7.2.1 Integer Programming (IP) Model

The availability of additional resource turns our attention to the use of time-indexed formulations. The IP formulation is based on time-discretization where time is divided into equal time periods (or time slots), where period t starts at time t and ends at time $t + 1$. Recall that T denotes the scheduling horizon, thus we consider time periods 0, 1, 2, …, T-1. At each discrete time slot, the total use of additional resource(s) should not exceed the available capacity. Hooker (2005, 2006) has stated that discrete time formulations are not only more straightforward but also perform more efficient than the continuous time formulations.

The following IP model is developed for the investigated problem case.

Indices:

i index of jobs (or dies) to be scheduled, $i = 1, ..., n$

j index of parallel machines, $j = 1, ..., m$

t index of time periods in the scheduling horizon, $t = 0, 1, …, T-1$

Parameters:

res_i the amount of operator required by job i.

b available number of operators for the scheduling horizon.

p_i processing time of job i.

M_i eligible machine set of job i.

Decision Variables:

$$x_{ijt} = \begin{cases} 1, & \text{if job } i \text{ begins its processing on machine } j \text{ at time } t. \\ 0, & \text{otherwise.} \end{cases}$$

$C_{\max}$ makespan, i.e., the maximum completion time of all jobs.

$$\min \; C_{\max}$$

subject to:

$$\sum_{j \in M_i} \sum_{t=0}^{T-1} x_{ijt}(t + p_i) \le C_{\max} \qquad i = 1, \ldots, n \qquad (7.1)$$

$$\sum_{i=1}^{n} \sum_{s=\max\{0, t-p_i\}}^{t-1} x_{ijs} \le 1 \qquad j = 1, \ldots, m \; ; \; t = 1, \ldots, T \qquad (7.2)$$

$$\sum_{j \in M_i} \sum_{t=0}^{T-1} x_{ijt} = 1 \qquad i = 1, \ldots, n \qquad (7.3)$$

$$\sum_{i=1}^{n} \sum_{j \in M_i} \sum_{s=\max\{0, t-p_i\}}^{t-1} \text{res}_i x_{ijs} \le b \qquad t = 1, \ldots, T \qquad (7.4)$$

$$x_{ijt} \in \{0, 1\} \qquad i = 1, \ldots, n \; ; j \in M_i \; ; \; t = 0, \ldots, T\text{-}1 \qquad (7.5)$$

The objective function aims to minimize makespan working with Constraints (7.1). Constraints (7.2) make sure that no more than one job can be assigned to any machine at any time period. Constraints (7.3) ensure that each job should certainly be processed on one of its eligible machines. Constraints (7.4) state that total number of operators assigned to machines at any time period is less than or equal to the available number of operators, b. Finally, all x_{ijt} are 0-1 variables as given in (7.5).

Van den Akker, Hurkens & Savelsbergh (2000) have stated that time-indexed formulations provide strong bounds by the use of its linear programming (LP) relaxation in comparison to any other alternative MIP formulations. Furthermore,

when side constraints are not tight, (e.g., in case of larger values of b in Constraints (7.4)), the problem can be more effectively solved with IP solvers utilizing problem dependent relaxations and cutting planes.

On the other hand, time-indexed formulations have a disadvantage with their size. The above formulation includes at most $n \cdot m \cdot T$ decision variables and $2 \cdot n + (m+1) \cdot T$ constraints. As the problem size gets larger, the solution time and the memory requirements will be larger.

A significant issue in solving combinatorial optimization problems is to find out tight lower bounds on the objective function. For the considered problem, we may use three lower bound schemes:

- machine load-based static lower bound (LB_{mach})
- operator load-based static lower bound (LB_{oper})
- machine load-based dynamic lower bound ($LB_{mach\text{-}dynmc}$)

The definition of first lower bound, LB_{mach}, comes from the classical PMS literature. Removing the machine eligibility constraints, (i.e., allowing all machines to process all jobs) and allowing the preemption of jobs, the following formulation, originally defined by McNaughton (1959), can be used to calculate LB_{mach}:

$$LB_{mach} = \max\left[\sum_{i=1}^{n} p_i / m, \max_{i=1,\ldots n}\{p_i\}\right] \tag{7.6}$$

The second lower bound, LB_{oper}, is related to the additional resource (i.e., operator). The average load per operator cannot exceed the final makespan value. Therefore, operator-load based lower bound is calculated as follows:

$$LB_{oper} = (\sum_{i=1}^{n} res_i p_i) / b \tag{7.7}$$

These two lower bounds are static and independent of IP formulation given above. Therefore, maximum of these two lower bounds, denoted as LB_1, is considered as a lower bound on the makespan:

$$LB_1 = \max\{LB_{mach}, LB_{oper}\} \tag{7.8}$$

Finally, the idea behind the machine load-based dynamic lower bound is as follows: The IP formulation given above also aims to determine job-machine assignments during its solution process. Therefore, sum of processing times of jobs assigned to any machine cannot exceed the final makespan value. The load of each machine *j* is then expressed as follows:

$$LBM_j = \sum_{i=1}^{n}\sum_{t=0}^{T-1} x_{ijt}\, p_i \qquad j = 1, \ldots, m \tag{7.9}$$

Accordingly, the maximum load among all machines determines machine-load based dynamic lower bound:

$$LB_{mach-dynmc} = \max_{j=1,2,\ldots m} LBM_j \tag{7.10}$$

Notice that, $LB_{mach\text{-}dynmc}$ contains decision variables x_{ijt} whose values are to be determined during the solution process of IP formulation. This is why we call it as a dynamic lower bound.

These two lower bound schemes are integrated into the entire IP model with the following constraints:

$$LB_1 \le C_{\max} \tag{7.11}$$

$$\sum_{i=1}^{n}\sum_{t=0}^{T-1} x_{ijt}\, p_i \le C_{\max} \qquad j = 1, \ldots, m \tag{7.12}$$

The computational runs show that adding these two constraints into the entire IP model provides very tight lower bounds and reduces the computation time significantly. However, the huge number of variables and constraints still may not allow us to reach the optimal and/or efficient results in a reasonable time.

7.2.2 CP Model

With the introduction of CP, a number of CP languages have been arisen. The formulation of CP model, in contrast to IP model, is highly dependent on the CP package used to model the problem because of the differences in constructs available in various modelling languages (Jain & Grossmann, 2001). In this study, ILOG's OPL Studio 3.7 (ILOG, 2003), which is originally developed by Van Hentenryck (1999), is used as the modelling language. This language supports the declarative representation of optimization problems, and includes the facilities to use both IP and CP.

A number of specialized scheduling objects are embedded into the OPL modeling language. A specialized solver, i.e., Scheduler 6.0 (ILOG, 2005c) is constructed to solve the problems that incorporate special scheduling objects. In this sub-section, the structures that have been used to model our investigated problem will be described. First, a special object "scheduleHorizon" is used to limit the domain of the search space. With this special object, the schedule length, T, is denoted. The basic OPL modeling framework involves a set of activities (i.e., jobs) that need to be completed using a set of resources (e.g., machines, operators, tools etc.). By default, an activity indicates a decision variable with three sub-variables, i.e., its *starting time*, its *duration* and its *ending time*, together with the constraints linking them. The set of jobs in our RCPMSP corresponds to the set of activities in this framework.

In OPL, a number of different resource types are available to capture the nature of the problem. A *unary resource* is a resource that cannot be shared by two activities at any time and a *discrete resource* is a resource that can be shared by several activities (ILOG, 2003). For the investigated problem, parallel machines can be modelled as unary resources (i.e., set of unary resources, S); whereas, operators can be treated as a discrete resource (defined as OPR with capacity of b) which can be shared by several activities. Total number of operators required by jobs at a given time cannot exceed the available number of operators, b.

The keyword "precedes" matches the ordinary precedence constraints. For example, if activity *a* should come before activity *b*, then the corresponding constraint with this keyword may simply be expressed as "*a* precedes *b*" which is equivalent to "a.end $\leq b$.start".

Similarly, another specific OPL object "requires" is the assurance of the disjunctive constraint that only one activity can be processed on a unary resource. Assume that a single machine (i.e., unary resource, *Machine*) should process *n* jobs, then the expression "*i* requires *Machine*, $i = 1, 2,\ldots n$" simply states that only one activity can be processed on unary resource *Machine* at a given time. Below disjunctive constraints are equivalent to this expression:

$$(i.\text{start} \geq i'.\text{end}) \vee (i'.\text{start} \geq i.\text{end}) \quad i = 1, 2,\ldots n\text{-}1, \quad i' = i+1, i+2,\ldots n.$$

Some other special objects are also available in OPL. More information is provided in Van Hentenryck (1999). OPL's special scheduling objects may contribute to the relatively good performance of CP with the built-in special purpose constraint propagation algorithms providing a more efficient domain reduction (see Kanet, Ahire & Gorman, 2004).

Using special constructs of OPL, the proposed CP model can be written as follows:

Minimize *makespan.end*

subject to:

i.duration $= p_i$ $\quad i = 1, \ldots, n$ (7.13)

i **precedes** *makespan* $\quad i = 1, \ldots, n$ (7.14)

i **requires** S $\quad i = 1, \ldots, n$ (7.15)

i **requires** (res_i) *OPR* $\quad i = 1, \ldots, n$ (7.16)

activityHasSelectedResource$(i, S, s_j) \Leftrightarrow assign_i = j \quad \forall i, j$ (7.17)

$$assign_i \neq j \qquad \forall i, j \mid j \notin M_i \qquad (7.18)$$

$$LB_1 \leq makespan.end \qquad (7.19)$$

$$\sum_{i=1}^{n} (assign_i = j)\, p_i \leq makespan.end \qquad j = 1, \ldots, m \qquad (7.20)$$

$$assign_i \in S \qquad i = 1, \ldots, n \qquad (7.21)$$

The objective function is defined as minimizing the completion time of dummy activity *makespan*. The duration of each job is defined with Equations (7.13). Constraints (7.14) ensure that completion time of any job should be smaller than the starting time of makespan activity**.** Constraints (7.15) enforce that job *i* needs a *unary resource S* (i.e., machines). Constraints (7.16) state that job *i* requires res_i units of discrete resource, *OPR* (i.e., operators). As explained earlier, **precedes** and **requires** are special OPL constructs. Constraints (7.17) use another OPL function **activityHasSelectedResource( )** that returns a value of true or false. These constraints ensure that if job *i* is processed using the *unary resource* corresponding to machine *j* ($s_j \in S$), then the subscript variable $assign_i$ is equal to *j*. Note that, $assign_i$ is a variable subscript and presents the machine selected to process job *i*. Constraints (7.18) state that if machine *j* is not compatible with job *i*, (i.e., $j \notin M_i$), the subscript variable, $assign_i$ cannot be equal to *j*. Constraint (7.19) and Constraints (7.20) correspond to lower bound schemes (i.e., LB_1, and $LB_{mach\text{-}dynmc}$) given in (7.11) and (7.12) respectively. Note that, $(assign_i = j)$ in (7.20) is a higher-order constraint which is widely utilized in most CP formulations. It returns true (1), when the equality is satisfied and returns false (0) otherwise. For each machine, sum of processing time of all jobs that satisfies the equality gives the load of that machine and accordingly gives a lower bound on the makespan. These two extra constraints strengthen the constraint propagation and accelerate the domain reduction process. Finally, (7.21) states that $assign_i$ takes values from the set of machine indices.

Notice that CP formulation given above, in comparison to IP model, has much fewer variables and constraints due to its declarative and compact framework.

As stated earlier, CP has an advantage on highly constrained problems. When side constraints complicate a problem (i.e., in case of fewer number of operators, *b*), CP can handle them and actually use them for an efficient constraint propagation and domain reduction (see e.g., Smith et al., 1997; Darbi-Dowman & Little, 1998; Focacci, Lodi & Milano, 2002; Milano & Trick, 2004). Another significant advantage of CP is its ability to define problem-based search procedures. Using an efficient search procedure in CP, we can prune the search tree in the earlier stages, and reach feasible solutions earlier. CP-based search procedures are discussed in Section 7.3. On the contrary, CP does not have a global view on the model and cannot utilize the advantage of relaxations. Optimization is tried to be done without using the objective function as a guide.

The following section presents a combined IP/CP OPL model (Van Hentenryck, 1999) that utilizes the complementary strengths of both IP and CP models.

7.2.3 Combined IP/CP OPL Model

Ideally, we would like to combine the strength of IP to handle the optimization part of the problem by using LP relaxation and the power of CP to find quick feasible solutions by using its better declarative framework (Jain & Grossmann, 2001). In such a combined approach, while the CP part of model focus on feasibility and tries to give quick feasible schedules, the IP part of the model tries to prove whether these feasible schedules are optimal (or not) using linear relaxation to produce lower bounds at each node of the search tree (Jain & Grossmann, 2001).

A combined IP/CP OPL model, (i.e., a double modelling framework discussed in Section 5.3.6) involves both IP and CP modelling variables and constraints. In addition, a number of linking constraints that present equivalence relations between the IP and CP variables are established (Van Hentenryck, 1999). Even though the size of the combined model is larger, it may still perform better because fewer nodes may have to be explored (Jain & Grossmann, 2001). The combined IP/CP model

proposed in this section has been motivated by the work of Jain & Grossmann (2001). The combined IP/CP OPL model for our RCPMSP can be written as follows:

min $C_{\max}$ **with linear relaxation**

subject to:

$$\sum_{j\in M_i}\sum_{t=0}^{T-1} x_{ijt}(t+p_i) \le C_{\max} \qquad i = 1, ..., n$$

$$\sum_{i=1}^{n}\sum_{s=\max\{0,t-p_i\}}^{t-1} x_{ijs} \le 1 \qquad j = 1, ..., m\ ;\ t = 1, \ldots, T$$

$$\sum_{j\in M_i}\sum_{t=0}^{T-1} x_{ijt} = 1 \qquad i = 1, ..., n$$

$$\sum_{i=1}^{n}\sum_{j\in M_i}\sum_{s=\max\{0,t-p_i\}}^{t-1} \mathrm{res}_i x_{ijs} \le b \qquad t = 1, \ldots, T$$

$$LB_1 \le C_{\max}$$

$$\sum_{i=1}^{n}\sum_{t=0}^{T-1} x_{ijt}\, p_i \le C_{\max} \qquad j = 1, ..., m$$

$$x_{ijt} \in \{0,1\} \qquad i = 1, ..., n\ ; j\in M_i\ ;\ t = 0, \ldots, T\text{-}1$$

(7.22)

i.duration $= p_i$ $\qquad i = 1, ..., n$

i **precedes** *makespan* $\qquad i = 1, ..., n$

i **requires** S $\qquad i = 1, \ldots, n$

i **requires** (res_i) *OPR* $\qquad i = 1, \ldots, n$

activityHasSelectedResource(i, S, s_j) $\Leftrightarrow$ $assign_i = j$ $\quad \forall i, j$

$assign_i \ne j$ $\qquad \forall i, j \mid j \notin M_i$

$LB_1 \le makespan.end$

$$\sum_{i=1}^{n}(assign_i = j)\, p_i \le makespan.end \qquad j = 1, \ldots, m$$

$assign_i \in S$ $\qquad i = 1, \ldots, n$

(7.23)

$$x_{i,assign_i,i.start} = 1 \qquad i = 1, ..., n$$

$makespan.end = C_{\max}$

(7.24)

This model is formulated by using the guidelines on basic framework of combined model structure in OPL. In this manner, the combined IP/CP model is an exact combination of the IP and CP models that were presented in the earlier sections. The solution algorithm for a combined IP/CP model primarily uses the CP solver. However, at each CP node, an extra LP relaxation, consisting of all the linear constraints in the combined model, is solved to obtain bounds for the objective function (Van Hentenryck, 1999). The objective function of the problem is the same as the one in the IP model. However, note that the objective function uses the keyword **with linear relaxation.** This keyword is necessary, since the model is no longer a pure integer program (e.g., it contains CP constraints) (Van Hentenryck, 1999). It makes sure that OPL uses the linear relaxation of all linear constraints on the objective function to determine a lower bound that can be used to prove optimality (ILOG, 2003). Constraint set (7.22) includes all IP constraints. Constraint set (7.23) includes all the constraints from the CP model. Finally, constraint set (7.24) links the IP and CP variables. The first constraint set in (7.24) links the x_{ijt} decision variables of IP with the $assign_i$ and i.start variables of CP for each job i. The second constraint gives the equivalence relations between the IP variable $C_{\max}$ and CP variable *makespan*.*end* in order to make the constraint propagation stronger.

In fact, there is no need to use all of the IP constraints in the combined IP/CP model. A common way is to keep the constraints which provide strong bounds to the objective function. However, our preliminary computational results show that keeping all of the IP constraints in the combined model provides best results in terms of constraint propagation and domain reduction. Therefore, all IP constraints are kept in the above formulation.

CP and IP/CP combined models may obtain more efficient results by introducing search procedures into the models. The following section discusses the use and necessity of search procedures in CP and proposes problem-based search procedures to be used in CP and IP/CP combined models.

7.3 CP-based Search Procedures

As stated earlier, by using an efficient search procedure in hard combinatorial optimization problems, we can prune the search tree in the earlier stages, and reach feasible solutions in advance (ILOG, 2003; Van Hentenryck, Perron & Puget, 2000). Henceforth, incorporating special-purpose search algorithms in the model may provide significant improvements in performance. OPL Studio 3.7 offers the ability to define search procedures in CP and IP/CP combined models.

A search procedure in OPL starts with the keyword `search`. It often consists of assigning values to variables. Such an assignment procedure generally chooses which variable to instantiate next (variable ordering) and then chooses which value to assign to the selected variable (value ordering). This process is repeated until all variables are instantiated. It is often critical to choose carefully which variable to instantiate next, since this choice specifies the size and the shape of the search tree (ILOG, 2003). Typically, variable and value ordering are implemented in OPL using the `forall` and `tryall` instructions, respectively (Van Hentenryck, Perron & Puget, 2000).

On the other hand, several built-in search procedures are available in OPL. Hooker (2005, 2006) states that setTimes (ILOG, 2003) and assignAlternatives (ILOG, 2003) search options in OPL specify a branching method that results in substantially better performance than the default method in some class of scheduling problems. The option setTimes is useful in scheduling problems with discrete resources and activities with fixed duration. In setTimes option, at first, OPL assigns starting time to all activities, and chooses an activity that can be scheduled at the earliest starting time *d*. It then decides whether to schedule the activity at time *d* or to postpone the activity. The process is then repeated for all activities that are not yet scheduled or not postponed. A postponed activity is reconsidered whenever its starting date is updated (ILOG, 2003). On the other hand, when alternative resources are used in a model, each activity using the alternative resources must be assigned a

resource from its set of unary resources (i.e., machines). OPL supports this by using a non-deterministic instruction assignAlternatives (ILOG, 2003).

The proposed CP and combined IP/CP models with these two built-in search options are solved in a range of test problems to evaluate their performances. The computational results related to these search options are given in Section 7.4.2. Using these options, we could not obtain satisfactory results in a large set of test instances. Therefore, problem-specific search procedures are proposed in this section.

While considering problem-based search procedures for our RCPMSP, there are two issues to focus on. Firstly, in the considered RCPMSP, some jobs have operator requirements through their processing time. However, a limited number of operators are available along the scheduling horizon to satisfy these resource requirements. Hence, once the jobs with non-zero operator requirements are assigned to machines and available time periods, it seems easy to allocate the jobs that require no operator to the remaining periods. The second significant factor is process flexibility of the parallel machines. Remember that, LFJ rule is often used for PMS problems with machine eligibility restrictions. The LFJ rule chooses the job that can be processed on the smallest number of machines first (Pinedo, 1995, p.71). With LFJ rule, the jobs with less machine alternatives are allocated to compatible machines in the earlier time periods, and the remaining jobs with more machine alternatives may then easily be allocated to one of their compatible machines with a probably small deterioration on the makespan value.

Considering above two issues, two problem based search procedures are proposed. The first one chooses the first variable (job) with highest amount of operator requirement. In case of ties, i.e., more than one job requires the same amount of operator(s), it selects the job with minimum number of machines in its domain set, i.e., apply LFJ rule. Such an assignment procedure reduces the domain size of the other variables through constraint propagation in the earlier stages of the search tree. The proposed search procedure is given in Figure 7.1. The function `dsize(variable)` denotes the domain size of the variable written in the parentheses.

```
search {
forall(i in Jobs ordered by decreasing <res_i,1/dsize(assign_i)>)
            {
            tryall(j in Machines: j∈M_i)
                  assign_i = j;
            tryall(t in 1..T)
                  i.start = t
                        onFailure
                  i.start ≠ t;
                              };
```

Figure 7.1 Proposed problem-based search procedure

Each iteration of `forall` instruction chooses another job that meets the conditions declared, and the process is iterated until all variables have been instantiated (or no solution was found). In case of value ordering, first `tryall` instruction assigns a value to the variable $assign_i$ from its eligible machines dynamically. The second `tryall` instruction assigns values to another variable `i.start` from its domain (i.e., range of discrete time periods in the schedule horizon, T). If at any point, one of the variables cannot be given a value consistent with the constraint store, it goes back to the previous variable and assigns another value to it. It should also be noted that, when the assignment `i.start = t` fails, the constraint `i.start ≠ t` is added to the constraint store before considering any alternative choice in the `tryall` instruction.

On the other hand, an alternative problem based search procedure can be proposed by reversing the priority of two issues in variable ordering procedure (see Figure 7.2). Notice that, in Figure 7.2, only the second line which gives the variable ordering procedure is different. The proposed reverse search algorithm first applies LFJ rule dynamically, and in case of ties, it chooses the job with higher operator requirement.

```
search {
forall(i in Jobs ordered by decreasing < 1/dsize(assign_i),res_i >)
            {
            tryall(j in Machines: j∈M_i)
                  assign_i = j;
            tryall(t in 1..T)
                  i.start = t
                        onFailure
                  i.start ≠ t;
                              };
```

Figure 7.2 Proposed reverse search procedure

Note that, all these four search procedures (setTimes, assignAlternatives, proposed search procedure, proposed reverse search procedure) are embedded into both CP and IP/CP OPL models. The detailed computational results related to these search procedures are presented in Section 7.4.2.

The next section presents the implementation issues and computational results for proposed optimization models.

7.4 Computational Results

This section firstly presents implementation issues with respect to problem parameters. Then, the generated problem instances are used to evaluate the performances of the search procedures given in Section 7.3. Finally, the computational results of IP, CP, and combined IP/CP models are presented and discussed.

7.4.1 Implementation Issues

The size of the optimization models for the investigated RCPMSP depends on the number of jobs, number of parallel machines and number of periods.

As stated earlier in Section 4.2, operator requirements of job strings are assumed to be fixed values, i.e. no operator, one-half, and one. Therefore, for all test problems, operator requirements, res_i, are generated with equal probability for the values 0, 0.5, and 1. Note that, for computational purposes, operator requirements have been made integer by multiplying them by two. The number of operators has also been accordingly increased.

To evaluate the proposed three optimization models, two main subsets of test instances are considered: 30 jobs, four machines, two operators; and 50 jobs, six machines, and three operators. The processing time, p_i, for each job is drawn from a uniform distribution on [4, 12] rounded to the nearest integer. There exist a number of reasons to choose these parameter levels for the problem. Firstly, the number of jobs and the number of machines are selected in such a way that approximately eight jobs are allocated to each machine. Secondly, the number of operators is determined with respect to number of jobs and number of machines in order that neither machine based static lower bound (LB_{mach}) nor operator-load based static lower bound (LB_{oper}) strongly dominates each other. In both subsets of test instances, these two lower bounds are valued so close to each other that the effect of both machine eligibility restrictions and additional resource constraints can be included and analyzed. Finally, the levels of all parameters are selected in such a way to generate medium sized test instances.

A significant issue for this class of problems is to determine an initial feasible period, T. For this purpose, an initial feasible heuristic (IFH) is developed to determine an efficient feasible period that may provide a magnificent reduction in the problem size.

In IFH, the following notation is used:

J set of jobs

M set of machines

R_t the current usage of operators at time t

NM_i the number of eligible machines that job i can be processed on.

C_i the completion time of job i

The details of IFH are given below. Note that, the computation time of IFH is negligible.

Step 1. Arrange the jobs in set J in decreasing order of res_i. Re-arrange the jobs with the same res_i, in decreasing order of p_i. In case of ties again, re-arrange the jobs in non-decreasing order of NM_i, i.e., apply LFJ rule. It is assumed that, the jobs are indexed by their rank in the arranged list. Note that $[i]$ denotes the job with index i in the arranged list. Set $R_t = 0$, for all t; $x_{ijt} = 0$, for all i, j, t; set i=1, j=1, t=0.

Step 2. Assign job $[i]$ to machine j at time t if the following conditions are satisfied: (i) $R_s + res_{[i]} \le b, s = t,..,t + p_{[i]} - 1$ (ii) $j \in M_{[i]}$, (iii) $\sum_{l=1}^{n} x_{ljs} = 0$, $s = t,..,t + p_{[i]} - 1$. Update $x_{[i]jt} = 1$; $R_s = R_s + res_{[i]}$, $s = t,..,t + p_{[i]} - 1$; $C_{[i]} = t + p_{[i]} - 1$, $i = i + 1$ and go to Step 4; otherwise, go to next step.

Step 3. If $j \ge m$, set $t = t$+1, j=1, go to next step; otherwise, j=j+1. Go to Step 2.

Step 4. If i>n, $T = \max_{i \in J}\{C_i\}$, STOP; otherwise, set j=1, t=0.Go to Step 2.

On the other hand, another significant issue is the flexibility measures of machine environment. As explained in Section 2.4, Vairaktarakis & Cai (2003) proposed two flexibility measures: *process flexibility* (F_P) and *balance flexibility* (F_B). In this chapter, we use two levels (i.e., low and high) of F_P and F_B and evaluate the proposed models in a wide range of flexibility of the machines. For all test instances, the cases that have F_P between the interval [0.2, 0.4] are defined as low processing flexibility cases, whereas the others that have a F_P in interval [0.6, 0.8] are considered as high processing flexibility cases. In terms of balance flexibility measure, the low and high levels of F_B are not fixed and differ with respect to both number of jobs and the level of F_P. All parameter levels corresponding to each sub-group of test problems are presented in Table 7.1.

Table 7.1 Sub-groups of test problems with parameter levels

# of Jobs (n)	#of Machines (m)	# of Operators (b)	Process Flexibility (F_P)	Balance Flexibility (F_B)
30	4	2	Low ($0.2 \leq F_P \leq 0.4$)	Low ($0 \leq F_B \leq 1$)
			Low ($0.2 \leq F_P \leq 0.4$)	High ($3 \leq F_B \leq 4$)
			High ($0.6 \leq F_P \leq 0.8$)	Low ($0 \leq F_B \leq 1$)
			High ($0.6 \leq F_P \leq 0.8$)	High ($4 \leq F_B \leq 5$)
50	6	3	Low ($0.2 \leq F_P \leq 0.4$)	Low ($0 \leq F_B \leq 1$)
			Low ($0.2 \leq F_P \leq 0.4$)	High ($4 \leq F_B \leq 5$)
			High ($0.6 \leq F_P \leq 0.8$)	Low ($0 \leq F_B \leq 1$)
			High ($0.6 \leq F_P \leq 0.8$)	High ($5 \leq F_B \leq 6$)

Proposed three models are implemented in ILOG OPL Studio 3.7 (ILOG, 2003) that uses CPLEX 9.0 (ILOG, 2005a) to solve an IP, ILOG Solver 6.0 (ILOG, 2005b) and ILOG Scheduler 6.0 (ILOG, 2005c) to solve a CP, and all of them to solve a combined IP/CP model. Since the several data instances of three models are to be solved in computational experiments, for composing and controlling optimization models, the script language of OPL, i.e. OPLScript (ILOG, 2003) is utilized. More specifically, OPLScript is used to generate and modify data instances, to apply IFH, to solve the same problem instance for different optimization models and finally to report and format the output data.

All generated problems are implemented in a Core 2 Duo 2.2 GHz, 2 GB RAM computer. In order to limit the solving time, a 1000 second run-time limit has been set.

The following section evaluates the performance of the CP-based search procedures.

7.4.2 The Performance Evaluation of CP-based Search Procedures

In Section 7.3, the CP-based search procedures (i.e., setTimes, assignAlternatives proposed search algorithm, and proposed reverse search algorithm) have been

presented and discussed. To evaluate the performance of these algorithms, three test problems from each combination of problem parameters given in Table 7.1 are solved. The related computational results of CP and IP/CP combined models are presented in Table 7.2.

In Table 7.2, the first five columns record the problem parameters. The sixth column denotes the sample number. The remaining columns present value of objective function, CPU time, and the gap percent of CP and IP/CP combined models with corresponding search procedures. The gap percent gives the deviation percent between the current objective and the optimal makespan value. The average CPU time and gap percent values for 30-job instances, 50-job instances and overall instances are also provided. Note that the values with a '+' superscript for the average CPU time are computed using only the instances for which the corresponding solver does not time out. Similarly, the average gap percent values with '1' superscript are computed using only the instances for which the corresponding model gives feasible and/or optimal solutions. Finally, at the bottom of Table 7.2, the number of feasible and optimal solutions belonging to each search procedure is also presented.

The results show that, with setTimes option, CP and IP/CP combined models cannot obtain even feasible solutions in 18 and 17 out of 24 test problems, respectively. Other built-in option, assignAlternatives, on the other hand, provides relatively better performance. With assignAlternatives option, CP and IP/CP combined models are able to give feasible results in 22 and 21 test problems, respectively. CP and IP/CP combined model with the reverse proposed search procedure gives feasible results in 23 and 24 test problems. Finally, the proposed search procedure with either CP or IP/CP combined model gives feasible solutions in all 24 test problem instances.

Table 7.2. Comparison of CP-based search algorithms

n	*m*	*b*	F_P	F_B	Samp. No	CP												IP-CP COMBINED APPROACH											
						setTimes			assignAlternatives			Prop. Reverse Alg.			Proposed Algorithm			setTimes			assignAlternatives			Prop. Reverse Alg.			Proposed Algorithm		
						Obj. Value	CPU Time	Gap %	Obj. Value	CPU Time	Gap %	Obj. Value	CPU Time	Gap %	Obj. Value	CPU Time	Gap %	Obj. Value	CPU Time	Gap %	Obj. Value	CPU Time	Gap %	Obj. Value	CPU Time	Gap %	Obj. Value	CPU Time	Gap %
30	4	2	Low	Low	1	-	1000*	-	71	208.89	0.00	74	1000*	4.23	71	83.68	0.00	71	201.71	0.00	72	1000*	1.41	71	34.83	0.00	71	22.28	0.00
					2	77	1000*	6.94	-	1000*	-	74	1000*	2.78	74	1000*	2.78	77	1000*	6.94	-	1000*	-	74	1000*	2.78	72	109.51	0.00
					3	56	1000*	1.82	56	1000*	1.82	60	1000*	9.09	55	100.18	0.00	55	272.64	0.00	57	1000*	3.64	60	1000*	9.09	55	52.87	0.00
			Low	High	1	68	1000*	1.49	67	107.12	0.00	69	1000*	2.99	67	66.35	0.00	68	1000*	1.49	68	1000*	1.49	67	141.52	0.00	67	46.80	0.00
					2	61	1000*	1.67	61	1000*	1.67	61	1000*	1.67	61	1000*	1.67	60	38.15	0.00	62	1000*	3.33	60	24.29	0.00	60	19.48	0.00
					3	-	1000*	-	83	1000*	6.41	80	1000*	2.56	78	148.60	0.00	-	1000*	-	83	1000*	6.41	78	542.79	0.00	78	43.24	0.00
			High	Low	1	-	1000*	-	63	1000*	1.61	63	1000*	1.61	63	1000*	1.61	-	1000*	-	63	1000*	1.61	62	8.31	0.00	62	8.43	0.00
					2	-	1000*	-	60	0.25	0.00	61	1000*	1.67	60	0.03	0.00	-	1000*	-	60	34.59	0.00	60	10.33	0.00	60	9.20	0.00
					3	-	1000*	-	66	667.88	0.00	-	1000*	-	66	2.45	0.00	-	1000*	-	67	1000*	1.52	68	1000*	3.03	66	12.77	0.00
			High	High	1	75	1000*	8.70	69	248.07	0.00	72	1000*	4.35	69	0.88	0.00	75	1000*	8.70	70	1000*	1.45	69	50.45	0.00	69	23.20	0.00
					2	-	1000*	-	73	1000*	4.29	72	1000*	2.86	70	44.19	0.00	-	1000*	-	73	1000*	4.29	70	22.36	0.00	70	18.81	0.00
					3	-	1000*	-	67	11.62	0.00	69	1000*	2.99	67	201.18	0.00	-	1000*	-	68	1000*	1.49	67	56.49	0.00	67	25.31	0.00
Average						-	1000*	4.12[1]	-	207.30[+]	1.44[1]	-	1000*	3.34	-	71.95	0.50	-	170.83	2.86	-	34.59	2.42	-	99.04	1.24	-	32.66	0.00
50	6	3	Low	Low	1	-	1000*	-	74	1000*	2.78	74	1000*	2.78	73	1000*	1.39	-	1000*	-	77	1000*	6.94	72	567.42	0.00	72	518.43	0.00
					2	-	1000*	-	-	1000*	-	75	9.15	0.00	75	0.10	0.00	-	1000*	-	-	1000*	-	75	345.18	0.00	75	404.75	0.00
					3	-	1000*	-	68	1000*	3.03	68	1000*	3.03	67	1000*	1.52	-	1000*	-	-	1000*	-	66	167.61	0.00	66	150.38	0.00
			Low	High	1	73	1000*	5.80	72	1000*	4.35	69	0.12	0.00	69	0.11	0.00	71	1000*	2.90	72	1000*	4.35	69	509.60	0.00	69	519.92	0.00
					2	-	1000*	-	75	1000*	1.35	76	1000*	2.70	75	1000*	1.35	-	1000*	-	76	1000*	2.70	74	259.46	0.00	74	310.02	0.00
					3	-	1000*	-	75	1000*	1.35	75	1000*	1.35	78	1000*	5.41	-	1000*	-	76	1000*	2.70	75	1000*	1.35	74	669.64	0.00
			High	Low	1	-	1000*	-	72	152.24	0.00	74	1000*	2.78	74	1000*	2.78	-	1000*	-	73	1000*	1.39	72	112.74	0.00	72	112.19	0.00
					2	-	1000*	-	67	1000*	1.52	67	1000*	1.52	67	1000*	1.52	-	1000*	-	68	1000*	3.03	66	88.32	0.00	66	86.24	0.00
					3	-	1000*	-	71	216.98	0.00	71	0.12	0.00	71	36.11	0.00	-	1000*	-	73	1000*	2.82	71	163.59	0.00	71	188.41	0.00
			High	High	1	-	1000*	-	69	1000*	2.99	70	1000*	4.48	69	1000*	2.99	-	1000*	-	73	1000*	8.96	67	614.48	0.00	67	298.58	0.00
					2	-	1000*	-	70	373.15	0.00	70	17.95	0.00	73	1000*	4.29	-	1000*	-	73	1000*	4.29	70	132.37	0.00	70	137.61	0.00
					3	-	1000*	-	81	1000*	1.25	82	1000*	2.50	82	1000*	2.50	-	1000*	-	85	1000*	6.25	80	426.90	0.00	80	462.06	0.00
Average						-	1000*	5.80[1]	-	247.45[+]	1.69[1]	-	6.84[+]	1.76[1]	-	12.11[+]	1.98	-	1000*	2.90[1]	-	1000*	4.34[1]	-	307.06[+]	0.11	-	321.52[+]	0.00
Average (Overall)						-	1000*	4.40[1]	-	220.69[+]	1.56[1]	-	6.84[+]	2.52[1]	-	56.99[+]	1.24	-	170.83[+]	2.86[1]	-	34.59[+]	3.34[1]	-	213.45[+]	0.68	-	177.09[+]	0.00
# of Feasible Solutions (Overall)						6			22			23			24			7			21			24			24		
# of Optimal Solutions (Overall)						0			9			4			12			3			1			20			24		

* Run is aborted due to time-limit (1000 seconds) .

[+] Avg. CPU times are computed using only the instances for which the corresponding solver does not time out.

[1] Avg. gap percent is computed using only the instances for which the corresponding model gives feasible and/or optimal solutions.

In terms of optimal solutions, setTimes option, again, presents a poor performance. It cannot give any optimal solutions with CP model, and only three optimal solutions with IP/CP combined model. When the performances of CP and IP/CP combined models with assignAlternatives option are compared, CP model, surprisingly, performs better than the IP/CP combined model in terms of number of optimal solutions and average overall gap. CP gives nine optimal solutions with 1.56% overall average gap, while IP/CP combined model could obtain only one optimal solution with 3.34% overall average gap. IP/CP combined model with the proposed search algorithms, on the other hand, gives relatively better results than the ones of CP model, as expected. With the proposed reverse algorithm, the CP model gives only four optimal solutions with 2.52% average gap, while IP/CP combined model gives optimal solutions in 20 test problems and provides 0.68% average gap. Similarly, with the proposed search algorithm, CP model gives 12 optimal solutions with 1.24% average gap; while IP/CP combined model gives optimal solutions to all test problems and accordingly zero average gap.

Consequently, in terms of getting efficient results in general, the proposed search algorithms produce much better results than the built-in search options (i.e., setTimes and assignAlternatives). Notice that, these built-in search options of OPL present general-purpose search algorithms and do not take the specific properties of the scheduling problem at hand into account, while the proposed search algorithms do.

Remember that, the proposed search algorithm gives priority to the jobs with higher operator requirements, while the proposed reverse algorithm gives priority to the least flexible jobs. In the overall performance, for both CP and IP/CP combined models, the proposed search algorithm provides relatively better performance than the proposed reverse search algorithm in terms of both number of optimal solutions and average gap percent. Moreover, in terms of computation times, the proposed search algorithm also spends less time for solving the test problems optimally.

Once we have investigated that the proposed search algorithm generally dominates the other ones; in the next section, for solving CP and combined IP/CP

models, we use only the proposed search algorithm for the further analyses and comparisons.

7.4.3 Numerical Results

To study the behaviour and characteristics of the IP, CP and IP/CP combined models, 12 test problems for each combination of problem parameters presented in Table 7.1 are randomly generated and solved.

The computational results are given in Appendix D and summarized in Table 7.3. The first five columns of Table 7.3. record the problem parameters. Since the IFH algorithm may result in different T values for each generated test problem and T is one of the parameters that determines the problem size, the average T values of 12 test problems has also been given in the sixth column. The remaining columns present number of cases that the corresponding model reaches optimal results, average CPU time (in seconds) and average gap percent. The gap percent gives the deviation percent between the current solution and the minimum makespan value of all three optimization models. Remember that the average CPU time values with a '+' superscript are computed using only the instances for which the corresponding solver does not time out. All other CPU time values are the average of all 12 test problems. Similarly, the average gap percent values with '1' superscript are computed using only the instances for which the corresponding model gives feasible and/or optimal solutions. To clarify these representations, let us explain the fourth row of Table 7.3:

- IP gives optimal results in 9 out of 12 test problems. The statement "127.41^{+}" declares that the average computational time of these nine test problems is 127.41 seconds and IP solver times out (i.e., exceeds 1000 second run time limit) in the remaining three test problems giving only feasible solutions. The average gap value "0.27^{1}" states that in at least one test problem, IP solver could not obtain even a feasible solution within the time limit (see Appendix D1, High-High-Sample 4). The average gap percent value of 0.27 is computed using 11 test instances that give feasible solutions.

Table 7.3 Computational results

n	*m*	*b*	F_P	F_B	Avg. # of Periods (*T*)	IP			CP			IP/CP COMBINED		
						# of Opt.	Avg. Time (*s*)	Avg. Gap %	# of Opt.	Avg. Time (*s*)	Avg. Gap %	# of Opt.	Avg. Time (*s*)	Avg. Gap %
30	4	2	Low	Low	74.08	9	239.83^{+}	0.56	5	37.88^{+}	1.29	12	52.00	0.00
			Low	High	74.58	10	106.35^{+}	0.24	6	44.22^{+}	0.98	12	38.46	0.00
			High	Low	68.00	10	90.53^{+}	0.23	7	35.48^{+}	0.83^{1}	12	14.60	0.00
			High	High	72.42	9	127.41^{+}	0.27^{1}	10	37.47^{+}	0.23	12	17.61	0.00
***n* =30, *m* =4, *b*=2**					72.27	38	141.03^{+}	0.33^{1}	28	38.76^{+}	0.84^{1}	48	30.67	0.00
50	6	3	Low	Low	74.75	6	411.08^{+}	0.69	3	62.53^{+}	1.13	10	349.32^{+}	0.12
			Low	High	76.82	3	266.67^{+}	1.06^{1}	4	266.02^{+}	1.32	12	391.54	0.00
			High	Low	71.30	3	488.35^{+}	1.33^{1}	4	9.10^{+}	1.19	12	150.91	0.00
			High	High	75.92	8	638.23^{+}	0.46	5	13.48^{+}	1.23	12	202.44	0.00
***n* =50, *m* =6, *b*=3**					74.70	20	451.08^{+}	0.88^{1}	16	87.78^{+}	1.22	46	273.55^{+}	0.03
Overall					**73.48**	**58**	**296.06^{+}**	**0.60^{1}**	**44**	**63.27^{+}**	**1.03^{1}**	**94**	**152.11^{+}**	**0.01**

$^{+}$ Avg. CPU times are computed using only the instances for which the corresponding solver does not time out.

^{1} Avg. Gap % is computed using only the instances for which the corresponding model gives feasible (or optimal) solutions.

- CP gives optimal results in 10 out of 12 test problems. The statement "37.47^{+}" declares that the average computational time of these 10 test problems is 37.47 seconds and CP solver times out (i.e., exceeds 1000 second run time limit) in the remaining two problems giving only feasible results. The value of "0.23" is the average gap value of all 12 test problems.
- IP/CP combined model gives optimal results in all 12 test problems. The average computational time of these test problems is 17.61 seconds. Of course, the average gap value is 0.00.

Using the IP/CP combined model, 94 out of 96 test problems are solved to optimality, while IP and CP models give optimal results in only 58 and 44 problem instances, respectively. As already stated, IP/CP combined model utilizes the complementary strengths of both IP and CP models and gets much better results.

CP model gives quick and feasible results in 95 test problems; however it could not improve objective value in 51 test problems. In fact, it is an expected result, since CP has a great advantage in obtaining quick and feasible results but cannot improve them due to the absence of global constraints (such as IP constraints with continuous relaxations).

Computational results also indicate that, in 52 out of 96 test problems, the combined IP/CP model dominates both other two models in terms of CPU time and/or the value of objective function (see grey shaded cells in Appendix D). IP and CP models outperform the other two models in 13 and 30 test problems, respectively. 11 out of 13 test problems that IP model outperforms other two models are the members of the sub-groups with low processing flexibility, F_P. IP may handle this sub-group of problems since each job has fewer machine alternatives to be processed on. This situation provides fewer nodes to be explored in B&B tree. More specifically, in terms of objective values, IP/CP combined model provides better makespan values in 22 test problems (see bold values within grey shaded cells in Appendix D), while IP gives better makespan value in only one test instance, and CP could not provide better makespan values in any instance. Moreover, 18 out of 22 test problems that IP/CP combined model provides better makespan values belong to 50-job test instances.

Computational results also indicate that IP/CP combined model provides relatively better performance in high values of F_P. It reaches optimal solutions in all 48 test problems with high level of F_P. Moreover, it can achieve up to two or three orders of reduction in CPU time in this group of test problems in comparison to cases with low level of F_P. In higher values of F_P, IP model remains with more machine alternatives causing much more B&B nodes to be explored. On the other hand, since the operators remain as obvious restricted resources in higher values of process flexibility (due to more machine alternatives for each job), and the proposed search algorithm gives priority to restricted additional resource, IP/CP combined model easily handles this range of test problems.

A significant point is that, the performances of IP and CP get worse with the problem size. While IP provides 38 optimal solutions (and 0.33% gap) in 30-job test instances, it could provide optimal results in only 20 test instances (and 0.88% gap) of 50-job test problems. Similarly, CP model provides 28 (0.84% gap) and 16 (1.22% gap) optimal solutions for 30-job and 50-job test instances, respectively. On the other hand, IP/CP combined model presents a more consistent performance by providing 48 (0% gap) and 46 (0.12% gap) optimal solutions for 30-job and 50-job test instances, respectively.

The results also show that computation time of all three models increases as the problem scales, reflecting the growing complexity of the problem. In overall, the IP/CP combined model spends much less computation time than other two models.

A significant point is that, the flexibility balance index, F_B, seems to have no meaningful effect on either average gap percent or computation time through the sub-groups of test problems.

Consequently, the results generally suggest that the combined IP/CP model with the proposed search procedure could be very useful in solving the minimum makespan RCPMSPs efficiently for medium sized problems.

Finally, a sample instance (i.e., n = 30 jobs, F_P = Low, F_B = Low, Sample No: 5, see Appendix D1) is chosen to illustrate the resulting schedules of all three proposed models. Figure 7.3, Figure 7.4 and Figure 7.5 illustrate the Gantt charts belonging to the related results of IP, CP and IP/CP combined models, respectively. In all figures, each job is illustrated with its index and resource requirement in parentheses. For instance, 18(2) indicates the 18^{th} job with two units of resource requirement. The time interval based resource usage profiles are also provided in the figures. Note that, in Figure 7.3 and Figure 7.4 additional resource is not fully utilized in some time intervals, whereas additional resource is fully utilized in the optimal schedule given by IP/CP combined model (see Figure 7.5). Note that, an optimal schedule may not fully utilize the additional resource.

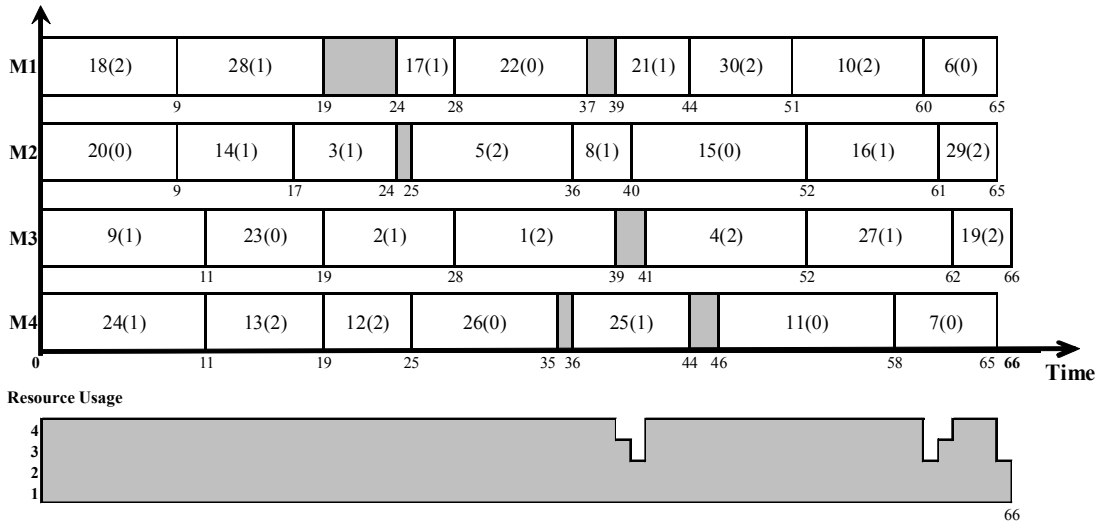


Figure 7.3 Resulting schedule of IP model for 30-Low-Low-Sample 5

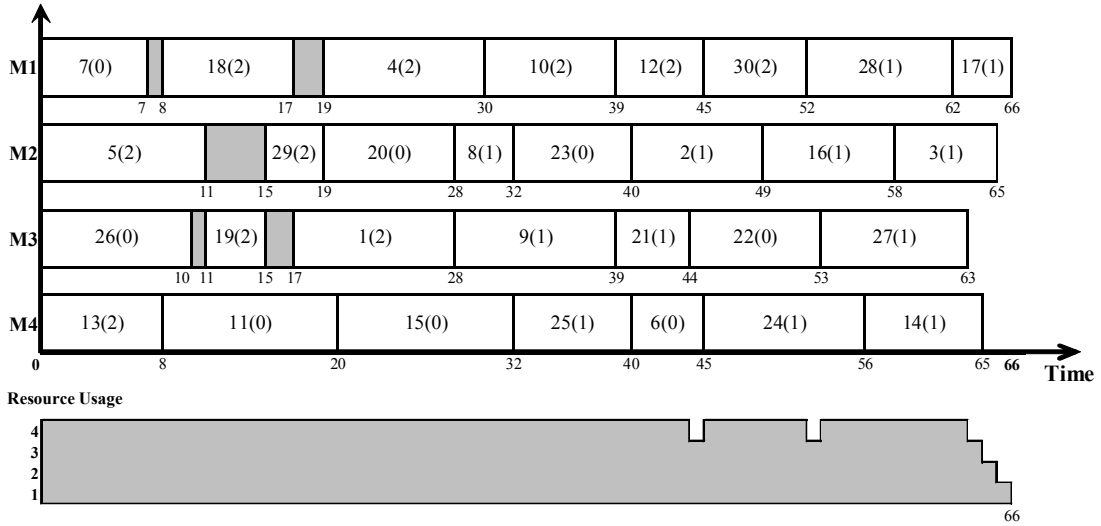


Figure 7.4 Resulting schedule of CP model for 30-Low-Low-Sample 5

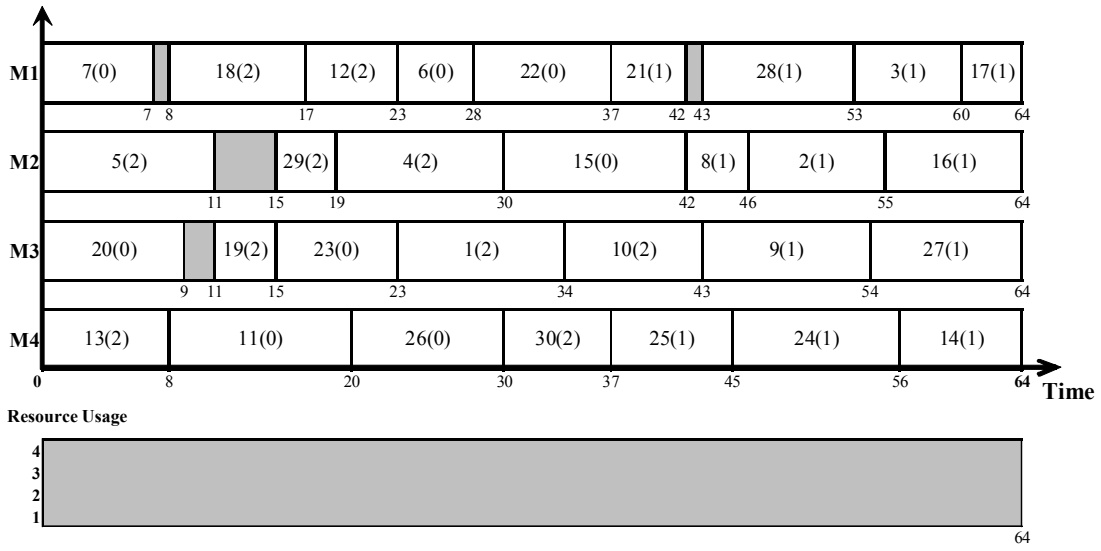


Figure 7.5 Resulting schedule of IP/CP combined model for 30-Low-Low-Sample 5

7.5 Chapter Summary

The proposed research in this chapter has addressed a RCPMSP with machine eligibility restrictions and arbitrary processing times. We have constructed three optimization models, namely, IP, CP, and combined IP/CP OPL model for this problem. Four different search algorithms have been evaluated and the one with the best performance, named proposed search algorithm, is embedded into the CP and IP/CP combined models. To determine the number of periods, which is a part of input for the three optimization models, a heuristic algorithm, named IFH, has also been developed.

The three optimization models are tested using 96 test problems which vary in number of jobs, number of machines, number of operators, process flexibility index (F_P), and balance flexibility index(F_B).

The computational results show that the IP/CP combined model with the proposed search algorithm outperforms the individual IP and CP models with respect to makespan values in almost all sub-groups of test instances. IP/CP combined model also achieves substantial reduction of computation times. It should also be noted that, in the high values of process flexibility (F_P), the combined IP/CP model provides a relatively better performance.

CHAPTER EIGHT
ITERATIVE SOLUTION APPROACHES FOR THE REAL CASE STUDY

8.1 Introduction

This chapter deals with the real case study i.e., P36$|res1\cdot 2, M_i, p_i|\ C_{\max}$ encountered in an injection molding department of an electrical appliance plant. Recall that it is a PMS problem with 36 machines, one additional resource type, arbitrary resource size and up to two units of resource requirements. It also includes machine eligibility restrictions and allows arbitrary processing times. The aim is to minimize makespan.

Firstly, the entire IP, entire CP and IP/CP combined models defined in Chapter 7 have been considered to solve the real case study. However, since the entire IP model has a huge number of variables for the real data, it may not handle the problem efficiently. On the other hand, although CP model has the ability of obtaining quick and feasible solutions, it may give a considerable deviation from the optima for the real data. And finally, although IP/CP combined model gives superior performance for medium size problems, it generally fails to find feasible and/or optimal solutions in large sized problems since its enormous number of IP and CP variables and constraints require a huge amount of computer memory.

Due to these restrictions, to obtain fast, efficient and practically applicable solutions for the real case problem, an iterative solution method which partitions the entire problem into loading and scheduling sub-problems is proposed. In the loading phase, an IP model is used to assign the jobs to machines with the aim of minimizing maximum load on machines. Subsequently, taking the dedicated jobs on each machine, two alternative models, namely, IP and CP have been built to construct the final schedule of the jobs with the objective of minimizing makespan subject to additional resource (operator) constraints. Indeed, once the job set of each machine is

determined by the loading phase, the problem in the scheduling phase is reduced to a RCPMSP with dedicated machines, which is discussed in Chapter 2 and 3.

The rest of this chapter is organized as follows. Section 8.2 introduces the considered problem. Section 8.3 presents the proposed iterative solution approaches i.e., IP/IP and IP/CP. The computational results are given and discussed in Section 8.4. Finally, Section 8.5 summarizes the chapter.

The following section represents the considered problem in detail.

8.2. Problem Statement

As stated in Chapter 4, the investigated problem is encountered in the injection-molding plant of an electrical appliance company. This plant produces several plastic parts for shipment to final assembly plants. The plant's objective is to meet weekly demand of parts subject to constraints on capacity, machine eligibility and operator availability.

The plant operates 12 hr per shift, two shifts a day, and five days a week. An MRP system explodes the dependent demand to generate weekly production orders for all parts. The managers of the plant aim to finish the manufacture of the current order quantities as soon as possible in order to leave some time and resource (machine and operator) capacity to the newly incoming orders that may occur along the current week. Henceforth, the objective function is chosen as the minimizing the makespan.

In the plant, each part has a die associated with it and its manufacture is completed on a single machine. There is only one die from each die type, and a die may be used to fabricate different parts. As already stated, the parts that share the same die are assumed to constitute a *job string*.

Consistent with the assumptions given in Section 4.2, the processing time of each job string (die) is determined as follows:

(a) Divide the weekly order quantity of each part by its hourly production rate to obtain the processing time of each part.
(b) Sum up the processing times of the parts that constitute a job string.
(c) Add one-hour setup time to this value.
(d) Finally, round up this value to the nearest hour. By rounding up to the nearest hour, other possible delays are incorporated into the total processing time of job strings.

Consequently, the main objective of this chapter is to develop a scheduling system that will receive MRP orders at the start of each week, and then generate the load of machines and the sequence of jobs with the objective of minimizing makespan.

The following section proposes an iterative solution approach that partitions the entire problem into loading and scheduling sub-problems.

8.3 Proposed Solution Approaches

PMS problems generally determine two kinds of decisions: job-machine assignment, and sequencing the jobs on the machines. The complexity usually grows exponentially with the number of machines (Graham et al., 1979). The presence of 36 machines in the real case problem adds a considerable complexity to the entire problem. Therefore, it may be advantageous to propose an iterative approach that partitions the entire problem into job-machine allocation (or loading) and sequencing (or scheduling) sub-problems.

Figure 8.1 illustrates the proposed solution approach. In the loading phase, an IP model assigns the jobs to the machines with the aim of minimizing maximum load on the machines. Subsequently, taking the dedicated jobs on each machine k (i.e., $JOBSET_k$) and maximum load value (i.e., $LOAD_{\max}$) as a lower bound on the makespan, two alternative models, namely, IP and CP are developed to construct the final schedule of the jobs with the objective of minimizing makespan.

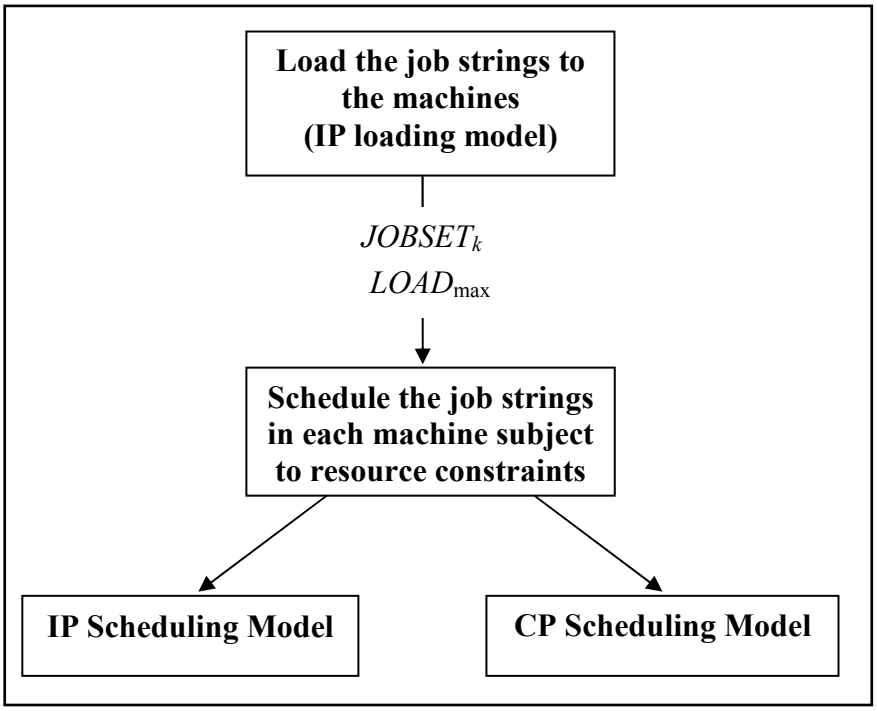


Figure 8.1 The summary of proposed solution methods

The following sub-sections explain the steps of the proposed algorithms in detail.

8.3.1 Loading Job Strings to Machines

The first sub-problem in the proposed solution approach deals with loading of job strings to machines using die-machine compatibility information.

In the considered problem, there is certainly some correlation between workload of machines and schedule makespan. In the loading phase, the workloads are balanced by minimizing the maximum load per machine, which is an obvious lower bound on the makespan objective of the scheduling problem. However, by balancing the workloads on machines, only a static measure of load balance is considered, whereas bottlenecks may appear dynamically in the scheduling phase due to the presence of additional resource constraints.

As stated earlier, IP may easily handle the assignment type problems. Therefore, in the loading phase a simple IP model is used to assign the jobs to the machines with the objective of minimizing maximum load over the machines. A similar IP model

has already been given by Vairaktarakis & Cai (2003) for PMS problems with machine eligibility restrictions. For IP loading model, binary decision variables a_{ik} and maximum load over the machines, $LOAD_{\max}$ are introduced:

$$a_{ik} = \begin{cases} 1, & \text{if job string } i \text{ is loaded on machine } k. \\ 0, & \text{otherwise.} \end{cases}$$

$LOAD_{\max}$ the maximum value of machine-based load.

min $LOAD_{\max}$

subject to:

$$\sum_{k \in M_i} a_{ik} = 1 \qquad i = 1, \ldots, n \qquad (8.1)$$

$$\sum_{i=1}^{n} p_i a_{ik} \le LOAD_{\max} \qquad k = 1, \ldots, m \qquad (8.2)$$

$$a_{ik} \in \{0, 1\} \qquad i = 1, \ldots, n \; ; \; k = 1, \ldots, m \qquad (8.3)$$

Constraints (8.1) ensure that a job string should be assigned to exactly one machine from its eligible machine set, M_i. Constraints (8.2) cover a similar idea of the machine based lower bound scheme indicated earlier and pick up the *maximum load*, $LOAD_{\max}$, over the machines working with the objective function. Finally, (8.3) states that all a_{ik} variables are 0 or 1.

In this IP model, some other secondary objective functions (e.g., minimizing total cost, $\sum_{i=1}^{n} \sum_{k=1}^{m} COST_{ik} a_{ik}$ where $COST_{ik}$ is the cost of assigning job i to machine k) may also be taken into consideration. In other words, IP model can be solved with respect to maximum load and total cost objective functions, hierarchically (see Edis & Ozkarahan, 2008). Nevertheless, since our main objective for the entire problem is minimizing makespan, we only deal with the objective of minimizing $LOAD_{max}$ in the loading phase.

8.3.2 Schedule the Job Strings

Since the IP model in the loading phase determines the set of job strings to be processed on each machine and finds out a $LOAD_{\max}$ value as a lower bound on the makespan objective, we define two additional parameters to be used in scheduling phase:

$JOBSET_k$: set of job strings to be processed on machine k. ($k = 1,\ldots,m$)

$\overline{LOAD}_{\max}$: lower bound on makespan value.

In the scheduling phase, two alternative models, namely, IP and CP are constructed with the aim of minimizing makespan. The following sub-sections give the details of these two models.

8.3.2.1 IP Scheduling Model

In this section, an IP model with time-indexed variables is introduced for the scheduling of job strings to minimize makespan. The decision variables and formulation of IP model are given below:

Decision Variables:

$$x_{it} = \begin{cases} 1, & \text{if job } i \text{ begins its processing at time } t. \\ 0, & \text{otherwise.} \end{cases}$$

$$\min \; C_{\max}$$

subject to:

$$\sum_{t=0}^{T-1} x_{it}(t + p_i) \le C_{\max} \qquad i = 1, \ldots, n \qquad (8.4)$$

$$\sum_{i \in JOBSET_k} \sum_{s=\max\{0,t-p_i\}}^{t-1} x_{is} \le 1 \qquad k = 1, \ldots, m \; ; \; t = 1, \ldots, T \qquad (8.5)$$

$$\sum_{t=0}^{T-1} x_{it} = 1 \qquad i = 1, \ldots, n \qquad (8.6)$$

$$\sum_{i=1}^{n} \sum_{s=\max\{0,t-p_i\}}^{t-1} res_i x_{is} \le b \qquad t=1, \ldots, T\text{-}1 \qquad (8.7)$$

$$\sum_{i=1}^{n} p_i res_i \le C_{\max} b \qquad (8.8)$$

$$\overline{LOAD}_{\max} \le C_{\max} \qquad (8.9)$$

$$x_{it} \in \{0,1\} \qquad i = 1, \ldots, n \; ; t= 0, \ldots, T\text{-}1 \qquad (8.10)$$

Since the job set of each machine, i.e., $JOBSET_k$, is known in advance, there is no need to use machine indices. The elimination of machine indices significantly reduces the number of variables in comparison to entire IP model. This IP scheduling model has $n \cdot T$ decision variables. Therefore, the problems with more jobs may become solvable. Constraints (8.4) pick up the maximum completion time of job strings. Constraints (8.5) ensure that no more than one job can be assigned to any machine at any time interval. Constraints (8.6) ensure that each job should certainly be processed. Constraints (8.7) guarantee that the number of operators assigned at each time unit is within its limit, *b*. Constraint (8.8) represents operator based lower bound scheme discussed in Section 7.2.1 The above formulation also includes a lower bound (i.e., $\overline{LOAD}_{\max}$) on the makespan value, which has been transferred from the loading phase. Constraint (8.9) covers this issue. Since these lower bounds narrow the solution space, the time to reach the optimal solutions significantly decreases. Finally, (8.10) states that all x_{it} are binary.

The iterative solution approach which uses the IP loading model and the proposed IP scheduling model is named as IP/IP iterative approach.

The following section presents the CP model to be used alternatively in the scheduling phase. The iterative approach which uses the IP loading model and the proposed CP scheduling model in the scheduling phase is going to be named as IP/CP iterative approach.

8.3.2.2 CP Scheduling Model

The same scheduling problem can also be modelled using CP. As in the previous chapter, ILOG's OPL Studio 3.7 (ILOG, 2003) is used as the modelling language.

Once the values of binary decision variables a_{ik} have been determined in the loading phase with IP model, we are able to define the machine index of each job. Therefore, a new parameter, $machine_i$ for the CP scheduling model is defined as follows:

$$(a_{ik} = 1) \Rightarrow machine_i = k \qquad i = 1,\ldots,n; \quad k = 1,\ldots,m. \tag{8.11}$$

Using the basic framework of OPL, CP scheduling model can be written as follows:

Minimize *makespan.end*

subject to:

$$i.\text{duration} = p_i \qquad i = 1, \ldots, n \tag{8.12}$$

$$i \ \textbf{precedes} \ makespan \qquad i = 1, \ldots, n \tag{8.13}$$

$$i \ \textbf{requires} \ machine_i \qquad i = 1, \ldots, n \tag{8.14}$$

$$i \ \textbf{requires} \ (res_i) \ OPR \qquad i = 1, \ldots, n \tag{8.15}$$

$$makespan.end \geq \overline{LOAD}_{\max} \tag{8.16}$$

$$(\sum_{i=1}^{n} i.duration \times res_i) / b \leq makespan.end \tag{8.17}$$

The objective function aims to minimize the completion time of dummy activity makespan. The duration of jobs are defined with Equations (8.12). Constraints (8.13) specify that completion time of any job should be less than or equal to the starting time of makespan activity. Constraints (8.14) enforce that job i needs the unary resource $machine_i$ which has already been determined for each job. Constraints (8.15) state that job i requires res_i units of additional discrete resource, OPR. Finally, similar to Constraint (8.9) in IP scheduling model, Constraint (8.16) states that

completion time of makespan activity should be greater than or equal to the machine-based lower bound, $\overline{LOAD}_{\max}$. Finally, (8.17) gives operator based-lower bound. These two lower bounds accelerate the constraint propagation and domain reduction in CP search tree.

As stated earlier, the setTimes option of OPL is useful in scheduling problems with discrete resources and activities with fixed duration. In the proposed iterative approach, since the loading phase makes the assignment of jobs to the machines, the only remaining issue is the scheduling of jobs subject to the available number of operators which can be treated as a discrete resource. Therefore, in CP scheduling model of the IP/CP iterative approach, setTimes search option has been used. The preliminary computations also show that it produces satisfactory results.

Recall that, the iterative solution approach which uses IP model in the loading phase and CP model in the scheduling phase is named as IP/CP iterative approach.

8.4 Computational Results

Since the entire IP and entire CP models given in Chapter 7 are also applicable to the real case problem, the results of these models are also provided in this sub-section. The analysis in Chapter 7 has shown that assignAlternatives and proposed search procedure perform more efficient than the other search procedures for the entire CP model. Therefore, entire CP model with these two search alternatives are also considered in the current computational studies. Since IP/CP combined model has failed to find feasible and/or optimal solutions for the real problem case, it has not been considered in computational analysis.

The entire IP model and entire CP model (with two search alternatives) given in Section 7.2 as well as proposed iterative solution approaches, i.e., IP/IP and IP/CP have been applied to test problems with 80 and 120 job strings (dies) and 36 machines with respect to real scheduling environment. The job strings to be processed are randomly selected from the list of 374 dies.

While generating the set of instances, in terms of machine eligibility restrictions, two levels of processing flexibility for the machines are considered with respect to process flexibility index (F_P) defined by Vairaktarakis & Cai (2003). Analyzing the die-machine compatibility information in the plant shows that the value of process flexibility (F_P) varies between 0.22 and 0.32. Therefore, for all test instances, the cases that have F_P in interval [0.22, 0.24] are defined as relatively low processing flexibility cases, whereas the others that have a F_P in interval [0.30, 0.32] are considered as relatively high processing flexibility cases.

The processing time, p_i, for each job string is drawn from a uniform distribution on interval [5, 25] with respect to corresponding interval of real demand values and then rounded to the nearest integer.

The number of operators available is taken as 10 and 12 with regard to real scheduling environment. The resource requirement of each job, res_i, is also taken as real data which is valued as 0, 0.5 or 1. For computational purposes, operator requirements have been made integer by multiplying them by two. The number of operators has also been accordingly increased.

For each combination of parameters, four test problems are solved. IP and CP models are implemented in ILOG OPL Studio 3.7 (ILOG, 2003). The proposed i.e., IP/IP and IP/CP iterative approaches are coded in OPLScript (ILOG, 2003).

All models are implemented on an Intel Core 2 Duo 2.2 GHz 2 GB RAM computer. A 1000-second run time limit is set to all models for each test instance.

The computational results are given in Table 8.1 and Table 8.2 for 80-job and 120-job test problems, respectively. The first four columns record the problem parameters. We also present objective function value, CPU time and gap percent of entire IP model and entire CP model with two search procedures. Since the proposed iterative approaches may use IP or (alternatively) CP model in the scheduling phase, the results of both IP and CP scheduling models are presented in these tables.

Table 8.1 Computational results (80 jobs)

# of Jobs (n)	Proc. Flex. (Fp)	# of Oper. (b)	Fp	Test Prob.	Entire IP Model				Entire CP Model (AssignAlternatives)				Entire CP Model (Proposed Search)				The Proposed Iterative Approaches								
																	Loading Phase		Scheduling Phase						
																			IP			CP			
					LB	C_{max}	CPU Time(s)	Gap %	C_{max}	CPU Best	CPU Time(s)	Gap %	C_{max}	CPU Best	CPU Time(s)	Gap %	$LOAD_{max}$	CPU Time(s)	C_{max}	CPU Time(s)	Gap %	C_{max}	CPU Best	CPU Time(s)	Gap %
80	Low	10	0.239	1	84	87	1000*	3.57	86	6.41	1000*	2.38	86	10.35	1000*	2.38	50	38.16	85	1000*	1.19	85	5.56	1000*	1.19
			0.230	2	84	85	1000*	1.19	85	15.62	1000*	1.19	87	6.95	1000*	3.57	42	14.89	86	1000*	2.38	85	9.15	1000*	1.19
			0.229	3	73	76	1000*	4.11	77	6.79	1000*	5.48	77	6.99	1000*	5.48	65	5.72	75	1000*	2.74	74	597.74	1000*	1.37
			0.235	4	78	79	1000*	1.28	80	6.55	1000*	2.56	82	6.86	1000*	5.13	53	10.96	**78**	381.08	0.00	**78**	5.55	5.55	0.00
		12	0.239	1	70	73	1000*	4.29	72	7.75	1000*	2.86	73	7.29	1000*	4.29	50	38.16	72	1000*	2.86	71	6.43	26.22	1.43
			0.230	2	70	72	1000*	2.86	73	8.18	1000*	4.29	73	6.99	1000*	4.29	42	14.89	**70**	335.63	0.00	71	5.96	1000*	1.43
			0.229	3	65	**65**	618.92	0.00	**65**	7.44	7.44	0.00	66	7.55	1000*	1.54	65	5.72	**65**	40.94	0.00	**65**	86.26	86.26	0.00
			0.235	4	65	67	1000*	3.08	66	9.81	1000*	1.54	68	7.85	1000*	4.62	53	10.96	**65**	517.63	0.00	**65**	227.90	227.90	0.00
	High	10	0.319	5	63	68	1000*	7.94	65	13.23	1000*	3.17	66	8.65	1000*	4.76	48	39.75	64	1000*	1.59	64	26.22	1000*	1.59
			0.310	6	76	85	1000*	11.84	77	849.84	1000*	1.32	79	706.17	1000*	3.95	63	65.61	77	1000*	1.32	**76**	11.74	11.74	0.00
			0.302	7	68	**68**	550.04	0.00	71	7.91	1000*	4.41	71	8.01	1000*	4.41	54	65.85	**68**	108.81	0.00	69	5.77	1000*	1.47
			0.309	8	72	74	1000*	2.78	76	378.07	1000*	5.56	75	7.19	1000*	4.17	50	8.94	**72**	218.32	0.00	73	7.03	1000*	1.39
		12	0.319	5	53	85	1000*	60.38	55	8.53	1000*	3.77	58	8.82	1000*	9.43	48	39.75	54	1000*	1.89	**53**	28.89	28.89	0.00
			0.310	6	63	-	1000*	-	64	15.06	1000*	1.59	66	8.54	1000*	4.76	63	65.61	65	1000*	3.17	64	47.80	1000*	1.59
			0.302	7	56	-	1000*	-	58	9.83	1000*	3.57	64	8.97	1000*	14.29	54	66.14	**56**	229.51	0.00	57	23.43	1000*	1.79
			0.309	8	60	66	1000*	10.00	62	12.02	1000*	3.33	63	8.27	1000*	5.00	50	8.77	**60**	109.31	0.00	**60**	11.11	11.11	0.00

1000.00*: Run is interrupted due to 1000-second run-time limit is exceeded.

Gray shaded cells: The corresponding model provides the best makespan value.

Values in Bold: The corresponding model gives the optimal makespan value.

Table 8.2. Computational results (120 jobs)

# of Jobs (n)	Proc. Flex. (Fp)	# of Oper. (b)	Fp	Test Prob.	Entire IP Model				Entire CP Model (AssignAlternatives)				Entire CP Model (Proposed Search)				The Proposed Iterative Approaches							
																	Loading Phase		Scheduling Phase					
																			IP			CP		
					LB	C_{max}	CPU Time(s)	Gap %	C_{max}	CPU Best	CPU Time(s)	Gap %	C_{max}	CPU Best	CPU Time(s)	Gap %	$LOAD_{max}$	CPU Time(s)	C_{max}	CPU Time(s)	Gap %	C_{max}	CPU Best	CPU Time(s)	Gap %
120	Low	10	0.239	9	101	-	1000*	-	103	9.83	1000*	1.98	103	8.42	1000*	1.98	62	6.79	103	1000*	1.98	102	5.87	1000*	0.99
			0.235	10	114	-	1000*	-	118	6.32	1000*	3.51	116	6.50	1000*	1.75	68	125.34	-	1000*	-	**114**	8.06	8.06	0.00
			0.239	11	115	-	1000*	-	119	6.04	1000*	3.48	118	13.98	1000*	2.61	82	5.43	-	1000*	-	116	6.51	1000*	0.87
			0.234	12	113	-	1000*	-	117	6.46	1000*	3.54	115	7.69	1000*	1.77	67	81.84	116	1000*	2.65	**113**	20.80	20.80	0.00
		12	0.239	9	84	-	1000*	-	86	9.68	1000*	2.38	87	10.73	1000*	3.57	62	6.79	90	1000*	7.14	85	6.39	1000*	1.19
			0.235	10	95	-	1000*	-	96	9.22	1000*	1.05	97	9.84	1000*	2.11	68	125.34	98	1000*	3.16	96	5.35	1000*	1.05
			0.239	11	96	-	1000*	-	101	8.33	1000*	5.21	99	8.91	1000*	3.13	82	5.43	-	1000*	-	97	7.41	1000*	1.04
			0.234	12	94	-	1000*	-	97	9.36	1000*	3.19	96	10.33	1000*	2.13	67	81.84	101	1000*	7.45	95	7.62	1000*	1.06
	High	10	0.301	13	106	-	1000*	-	110	7.34	1000*	3.77	108	8.35	1000*	1.89	78	8.73	-	1000*	-	108	162.66	1000*	1.89
			0.304	14	106	-	1000*	-	115	6.60	1000*	8.49	108	8.37	1000*	1.89	84	126.28	-	1000*	-	107	7.72	1000*	0.94
			0.305	15	94	-	1000*	-	98	8.38	1000*	4.26	96	10.45	1000*	2.13	76	17.60	-	1000*	-	95	5.57	1000*	1.06
			0.304	16	100	-	1000*	-	102	10.01	1000*	2.00	102	110.31	1000*	2.00	81	7.72	111	1000*	11.00	101	5.36	1000*	1.00
		12	0.301	13	89	-	1000*	-	91	9.63	1000*	2.25	99	9.60	1000*	11.24	78	8.73	94	1000*	5.62	**89**	32.00	32.00	0.00
			0.304	14	89	-	1000*	-	94	8.84	1000*	5.62	100	9.70	1000*	12.36	84	126.28	-	1000*	-	90	5.92	1000*	1.12
			0.305	15	78	-	1000*	-	86	10.42	1000*	10.26	92	11.46	1000*	17.95	76	17.60	85	1000*	8.97	79	246.64	1000*	1.28
			0.304	16	83	-	1000*	-	85	835.87	1000*	2.41	88	12.33	1000*	6.02	81	7.72	87	1000*	4.82	85	159.16	1000*	2.41

1000.00*: Run is interrupted due to 1000-second run-time limit is exceeded.

Gray shaded cells: The corresponding model provides the best makespan value.

Values in Bold: The corresponding model gives the optimal makespan value.

Note that, the gap percent gives the deviation percent between the current result and the lower bound (LB) of the entire IP model. Since CP is able to reach solutions quickly, a column that gives CPU time to reach corresponding best solution, i.e., “CPU best” is provided for entire CP models with two search procedures and the proposed IP/CP iterative approach.

The gray-shaded cells in these tables indicate the best results in terms of makespan values. Moreover, the makespan values in bold specify that the results are also optimal. As can be seen from these tables, the proposed IP/CP iterative approach dominates the entire IP and entire CP models in almost all sub-groups of test instances. The performance of IP/CP iterative approach is comparable with IP/IP iterative approach in 80 job test instances where IP scheduling model may handle the scheduling sub-problem due to presence of relatively less number of variables. On the other hand, in 120 job test instances, IP/CP iterative approach dominates the IP/IP approach. Therefore, CP scheduling model is relatively better in scheduling problems with more number of jobs. Furthermore, except a few test problems, IP/CP iterative approach reaches its best solution in less than a couple of minutes. It confirms that CP has an advantage to reach efficient results in advance and can be used to get quick and practical solutions in this type of problems.

The average gap percent values of each sub-group of eight problems with respect to number of jobs and number of operators are presented in Figure 8.2. As expected, IP/CP iterative approach gives the smallest average gap percent in all sub-groups of test problems. As stated earlier, the performances of IP/IP and IP/CP iterative approaches are comparable for 80-job test problems where the overall average gap percent values of IP/IP and IP/CP iterative approaches are 1.07 and 0.90, respectively. For 120-job test problems, the performance of IP/CP iterative approach is much better than IP/IP iterative approach since IP scheduling model could not provide efficient results due to increasing problem size.

Since IP is an appropriate technique in solving assignment type problems and CP is better in sequencing and scheduling problems, an iterative solution procedure

which uses IP in assigning the jobs to the machines and then uses CP in scheduling the jobs on the dedicated machines gives better performance than the other proposed solution approaches.

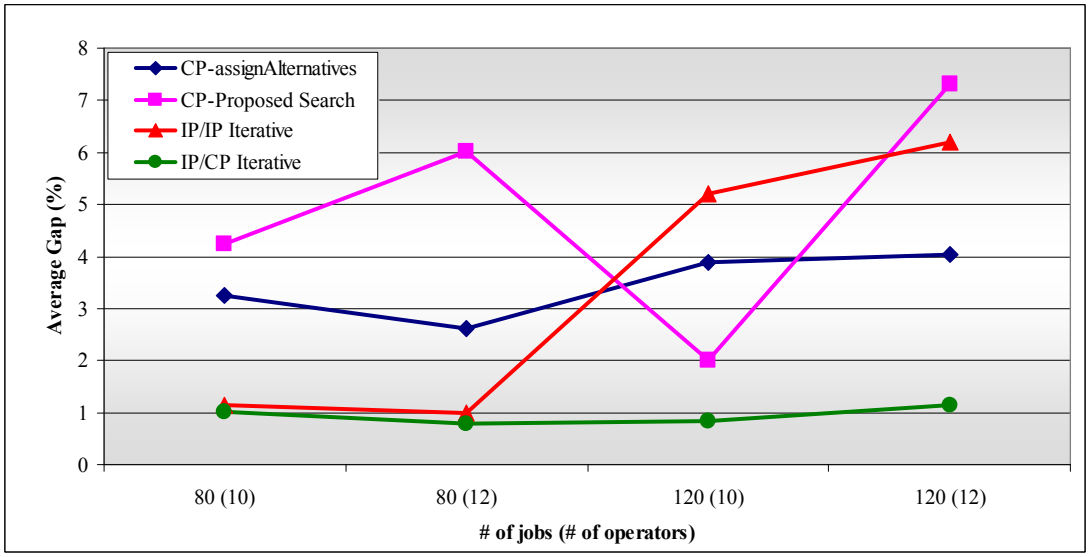


Figure 8.2 Comparison of solution approaches in terms of average gap (%)

In terms of CP models, the proposed search algorithm provides relatively better performance in the test problems with more jobs and less number of operators. More specifically, since the proposed search algorithm gives priority to the jobs with higher operator requirements, the best performance of CP model with proposed search procedure is encountered in 120 job test problems with 10 operators where the additional resource dominates the schedule (see Figure 8.2). On the contrary, assignAlternatives search option provides its best performance in 80 job test problems with 12 operators where the effect of additional resource constraints in constructing the schedule is relatively weak. In such cases, unary resources (i.e., machines) play a significant role in constructing the schedule. Therefore, the search option assignAlternatives, which deal with unary resources in a more efficient way, provides better makespan values.

Since the entire IP model could not give even feasible solutions for 120-job test problems within 1000 second run time limit, the analysis focuses on the computational results of 80-job test instances. The computational results related to

80-job test problems indicate that, the entire IP model may handle low processing flexibility cases where each job has fewer machine alternatives to be processed on. This situation provides fewer nodes to be explored in B&B tree. In fact, it is also the case for the entire CP models. The performance of entire CP model with both search procedures is relatively better in low processing flexibility cases. On the other hand, for the iterative solution methods, since the set of jobs assigned to each machine has already been determined in the loading phase, the performances of IP/IP and IP/CP scheduling models do not depend on the process flexibility variation of test problems.

Consequently, it can be stated that IP/CP iterative approach works efficiently for the real case problem in terms of both average gap percent and getting quick and efficient results.

Finally, a sample instance (i.e., n = 120, F_P = Low, b = 10, Test Prob.10, see Table 8.2) is chosen to illustrate the resulting optimal schedule of IP/CP iterative approach. The Gantt chart of this sample is presented in Figure 8.3. Similar to earlier representations, in Figure 8.3, each job is described with its index and resource requirement in parentheses. The time interval based resource usage profile is also provided in the figure. Note that, the utilization of operators is at its highest level in almost all time intervals, since the additional resource constraints dominate the schedule. On the other hand, since the number of operators is relatively small in comparison to number of jobs and process flexibility (F_P) is low, there exists some machine idle times in the schedule.

Figure 8.3 Resulting schedule of IP/CP iterative approach for n = 120, F_P = Low, b = 10, Test Prob.10

8.5 Chapter Summary

The problem considered in this chapter is motivated by a real-world RCPMSP problem investigated in the injection-molding department of an electrical appliance plant. An iterative solution method which partitions the problem into loading and scheduling sub-problems is proposed. In the loading phase, an IP model assigns the jobs to the machines with the aim of minimizing maximum load on machines. Subsequently, taking the dedicated jobs of each machine from the loading phase, two alternative scheduling models, namely, IP and CP, construct the final schedule of the jobs with makespan objective subject to operator constraints. Computational results show that IP/CP iterative method provides better makespan values for almost all test problems in comparison to entire IP and entire CP models. The performance of IP/IP iterative approach is comparable with IP/CP iterative approach in 80-job test instances; whereas, in 120-job test instances, IP/CP iterative approach dominates the IP/IP approach. Consequently, although the proposed IP/CP iterative approach may not give an optimal solution to the entire problem, it provides efficient results with very small optimality gaps in a reasonable amount of time.

CHAPTER NINE
CONCLUSIONS AND FUTURE RESEARCH

9.1 Summary

Scheduling models and algorithms are widely used in manufacturing applications to perform production in an efficient way. PMS is one of the most studied areas in the scheduling literature. In most of PMS studies, the only considered resources are machines. However, in most real life problems, jobs may also require certain additional resources, such as machine operators, tools, dies, pallets etc. for their handling and processing. Beside the additional resources, jobs may often not be processed on any of the parallel machines but rather must be processed on a machine belonging to a specific subset of machines. This situation, named machine eligibility restrictions, is also widely encountered in real scheduling environments.

Analyzing the related literature reveals that no study, so far, has considered additional resources and machine eligibility restrictions together for the PMS problems. The proposed research in this dissertation has been motivated by a real world PMS problem where additional resource constraints and machine eligibility restrictions should also be taken into consideration.

By the motivation of the real-life problem, three problem cases have been investigated in this dissertation. All three problems include machine eligibility restrictions (M_i) and one additional resource type with arbitrary resource size and up to two units of resource requirements (i.e., $res1 \cdot 2$) as the common characteristics. The representations of problem cases with their distinct characteristics are as follows:

- **Problem Case I:** $P \mid res1 \cdot 2, M_i, p_i=1 \mid \sum_i C_i$. Unit (equal) processing times with the aim of minimizing total flow time.
- **Problem Case II:** $P \mid res1 \cdot 2, M_i, p_i \mid C_{\max}$. Arbitrary processing times with the aim of minimizing makespan.

- **Problem Case III:** $P36\ |res1\cdot 2,\ M_i,\ p_i|\ C_{\max}$. The real case study of the second problem with 36 machines, and real die-machine compatibility matrix.

For the first problem case, two heuristic algorithms have been proposed. The first one is a Lagrangian-based solution approach (LSA) embedded into a subgradient optimization procedure to obtain tight lower bounds and near optimal solutions. The second one is an independent problem specific heuristic (PSH). By means of randomly generated instances of the problem, it has been shown that the proposed algorithms provide efficient results with a small optimality gap and LSA derives very tight lower bounds. Moreover, LSA gives superior results in low flexible machine environments, while PSH is relatively better in high flexible ones.

For the second problem case, three optimization models, namely, IP, CP, and combined IP/CP models have been developed. Four different search algorithms have been evaluated and the one with the best performance, named proposed search algorithm, is embedded into the CP and IP/CP combined models. The three optimization models are tested through randomly generated test problems. The IP/CP combined model with the proposed search algorithm has outperformed the individual IP and CP models with respect to makespan values in almost all sub-groups of test instances. IP/CP combined model has also achieved a considerable reduction in computation times. In addition, in the high values of process flexibility, the combined IP/CP model has provided much better performance.

For the real-life case study, IP/IP and IP/CP iterative approaches have been proposed. Iterative solution methods partition the entire problem into loading and scheduling phases. In the loading phase, an IP model assigns the jobs to the machines with the aim of minimizing maximum load on machines. Subsequently, in the scheduling phase, two alternative models, namely, IP and CP construct the final schedule of the jobs with makespan objective subject to resource constraints. Entire IP and entire CP models have also been applied to the test problems with real data. Computational results have shown that, IP/CP iterative approach outperforms the other approaches by providing quick and efficient results with small optimality gaps.

Figure 9.1 outlines the research problems with the proposed solution approaches.

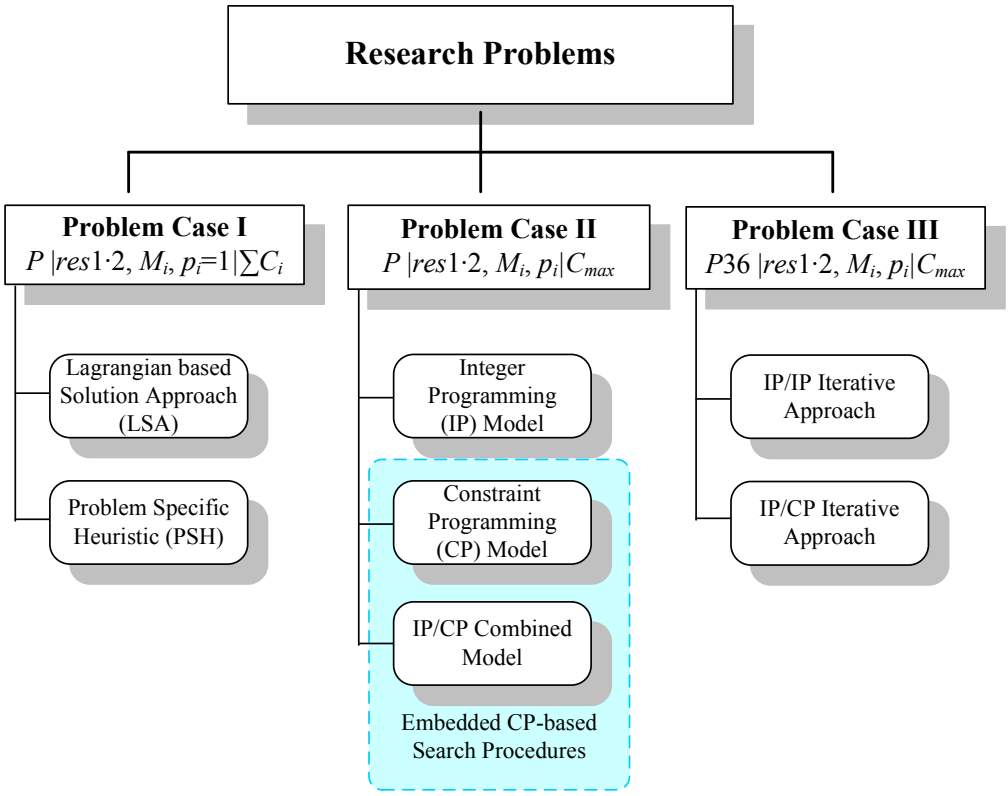


Figure 9.1 Research problems and proposed solution approaches

9.2 Contributions

In the literature related to RCPMSPs, although a number of studies handle common shared resources, most of them deal with dedicated (i.e., the set of jobs to be processed on each machine is priori known) or identical machines. As far as we know, no study in this field has considered machine eligibility restrictions.

All the research problems in this dissertation, differently from previous studies, consider machine eligibility restrictions and common shared resource (i.e., machine operators shared by all machines) cases together. This is one of the main contributions of this dissertation.

In case of machine eligibility restrictions, different flexibility measures of machines requires further analysis on their different levels. So far, the effect of these flexibility measures has only been discussed within classical PMS systems.

This dissertation analyzes the effect of machine eligibility restrictions for the research problems in terms of process flexibility and balance flexibility.

The studies related to RCPMSPs mainly focus on small sized problems with hypothetical data. Large sized problems, especially the cases encountered in real-life environments, do not receive much attention in the literature.

The research problems in this dissertation are motivated by a real RCPMSP with machine eligibility restrictions encountered in an injection molding department of an electrical appliance plant. Moreover, the third problem case also considers real case study with its own large sized real data (i.e., 36 machines, up to 120 jobs and 12 units of additional resource).

Although many researchers have studied the use of Lagrangian relaxation algorithms for PMS problems, to the best of our knowledge, only Ventura & Kim (2003) applied this technique for a RCPMSP with identical machines and unit processing times. However, they do not consider machine eligibility restrictions.

In this dissertation, a Lagrangian-based solution approach is proposed for the first problem case. The proposed solution approach not only provides tight lower bounds but also produces efficient results with small optimality gaps.

Since RCPMSPs are natural candidates for strict feasibility problems, CP technique may be utilized individually or as a part of a solution approach for this class of problems. To the best of our knowledge, no study so far utilizes CP technique for solving RCPMSPs.

CP technique is utilized in both second and third research problems to obtain quick and efficient results.

Although CP has an advantage of finding quick and feasible results, it usually lacks proving the optimality when it is used alone. Therefore, for the second problem case, an IP/CP combined model is developed to utilize the complementary strengths of two techniques. As far as we know, it is the first study that proposes an IP/CP combined model for RCPMSPs.

For the third problem case, we propose an iterative solution method which partitions the entire problem into loading and scheduling sub-problems to obtain more efficient results. The scheduling sub-problem is solved by IP and also alternatively by CP. The efficiency of CP is shown in the scheduling phase.

One of the advantages of CP is its ability to use search procedures. By using an efficient search procedure in CP, the search tree can be pruned in the earlier stages, and feasible solutions can be reached in advance. No study so far utilizes problem specific CP-based search algorithms in this class of problems.

For the second problem case, two problem specific CP-based search procedures have been proposed to be used in both CP and IP/CP combined OPL models. The efficiency of the proposed search procedures is also confirmed by comparing them with built-in search procedures of OPL optimization software.

9.3 Future Directions

The following issues remain for future research of RCPMSPs.

The proposed research on the first research problem can be extended to the problem cases with different objectives and non-equal processing times. In such problem cases, the remaining LRP may not be solved easily with an ordinary use of

Lagrangian relaxation. Therefore, suitable Lagrangian decomposition schemes may be developed to obtain more efficient results in such problem cases.

Since the combined IP/CP model developed for the second problem case gives optimal results for a wide range of test problems, this model can be suggested to be used for evaluating the performance of other solution approaches which will be proposed for this class of RCPMSPs.

The search procedures are very important for the CP part of the problem. The performance of the proposed search algorithms confirms the efficiency and success of problem-based search algorithms in comparison to general-purpose search options. Therefore, alternative problem-specific search procedures may reduce the computation time of both CP and IP/CP combined model in different types of RCPMSPs.

Since IP/CP combined model performs relatively better in the test problems with high process flexibility, the combined IP/CP model can successfully be adapted to the problems with identical parallel machines where all machines are capable of processing all jobs.

Although the proposed research focuses on a specific RCPMSP of a real-world scheduling environment, it can easily be extended to more general versions of RCPMSPs, e.g. unrelated parallel machines, arbitrary number of additional resources etc.

Since the additional resources in the research problems are common rather than the ones assigned to parallel jobs of each facility, the considered RCPMSP may not be easily decomposed into independent sub-problems. Nevertheless, if IP model is constructed in a different manner with different assumptions, such a decomposition approach (see e.g., Chu & Xia, 2005; Hooker 2005, 2006) may also be taken into consideration.

A further study might be consideration of exact scheduling of operators. Once the final schedule of jobs is obtained by proposed solution methods, an additional operator scheduling model may be constructed. Moreover, an integrated operator and machine scheduling framework may also be developed. Similar works in the literature are given by Artigues, Gendreau & Rousseau (2007) and Artigues, Gendreau, Rousseau & Vergnaud (2009) for a different scheduling environment.

Finally, computational results generally show that CP model may produce very quick and efficient results due to the strict feasibility structure of RCPMSPs. On the other hand, IP may require rather enormous amount of time to obtain even a feasible solution. Therefore, giving the quick solutions of CP to IP solvers as initial solutions may produce efficient results in less computation times.

REFERENCES

Artigues, C., Gendreau, M., & Rousseau, L-M. (2007). A flexible model and a hybrid exact method for integrated employee timetabling and production scheduling. In E.K. Burke & H. Rudov´a (Eds.). *PATAT 2006, Lecture Notes in Computer Science 3867*, (67–84). Germany: Springer-Verlag Berlin Heidelberg.

Artigues, C., Gendreau, M., Rousseau, L-M., & Vergnaud, A. (2009). Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. *Computers & Operations Research*, *36*, 2330-2340.

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *Constraint-based scheduling: Applying constraint programming to scheduling problems*. London, Great Britain: Kluwer Academic Publishers.

Baptiste, P., Laborie, P., Le Pape, C., & Nuijten,W. (2006). Constraint based scheduling and planning. In F. Rossi, P. Van Beek, & T. Walsh (Eds.) *Handbook of Constraint Programming* (761–799). Amsterdam: Elsevier.

Beasley, J.E. (1995). Lagrangean Relaxation. In C.R. Reeves, (Ed.). *Modern heuristic techniques for combinatorial problems* (243-303). UK: McGraw-Hill.

Blazewicz, J. (1979). Deadline scheduling of tasks with ready times and resource constraints. *Information Processing Letters*, *8*(2), 60-63.

Blazewicz, J., Barcelo, J., Kubiak, W., & Röck, H. (1986). Scheduling tasks on two processors with deadlines and additional resources. *European Journal of Operations Research*, *26*, 364-370.

Blazewicz, J., Brauner, N., & Finke, G. (2004). Scheduling with discrete resource constraints. In J. Y-T. Leung, (Ed). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis.* USA: CRC Press.

Blazewicz, J., Cellary, W., Slowinski, R., & Weglarz, J. (1986). *Scheduling under resource constraints – Deterministic Models*. Switzerland: J. C. Baltzer AG, Scientific Publishing Company.

Blazewicz, J., & Ecker, K. (1983). A linear time algorithm for restricted bin packing and scheduling problems. *Operations Research Letters*, *2*(2), 80-83.

Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., & Weglarz, J. (2007a). Communication Delays and Multiprocessor Tasks. In *Handbook on scheduling: From theory to applications* (199-241). New York: Springer-Verlag

Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., & Weglarz, J. (2007b). Scheduling under Resource Constraints. In *Handbook on scheduling: From theory to applications* (425-475). New York: Springer-Verlag Berlin Heidelberg.

Blazewicz, J., Kubiak, W., & Martello, S. (1993). Algorithms for minimizing maximum lateness with unit length tasks and resource constraints. *Discrete Applied Mathematics*, *42*, 123-138.

Blazewicz, J., Kubiak, W., Röck, H., & Szwarcfiter, J. (1987). Minimizing mean flow time with parallel processors and resource constraints. *Acta Informatica*, *24*, 513-524.

Blazewicz, J., Lenstra, J.K., & Rinnooy Kan, A.H.G. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics*, *5*, 11-24.

Bosch, R., & Trick, M. (2004). Constraint programming and hybrid formulations for three life designs. *Annals of Operations Research*, *130*(1-4), 41-56.

Bourland, K.E., & Carl, L.K. (1994). Parallel machine scheduling with fractional operator requirements. *IIE Transactions*, *26*(5), 56-65.

Brailsford, S.C., Potts, C.N., & Smith, B.M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, *119*, 557-581.

Brucker, P. (2004). *Scheduling algorithms* (5th ed.). New York: Springer-Verlag Berlin Heidelberg.

Centeno, R., & Armacost R.L. (1997). Parallel machine scheduling with release time and machine eligibility restrictions. *Computers and Industrial Engineering*, *33*(1-2), 273-276.

Centeno, R.,& Armacost R.L. (2004). Minimizing makespan on parallel machines with release time and machine eligibility restrictions. *International Journal of Production Research*, *42*(6), 1243-1256.

Chen, J-F. (2005). Unrelated parallel machine scheduling with secondary resource constraints. *International Journal of Advanced Manufacturing Technology*, *26*, 285-292.

Chen, J-F., & Wu, T-H. (2006). Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. *Omega*, *34*, 81-89.

Cheng, T.C.E., & Sin, C.C.S. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, *47*, 271-292.

Chu, Y., & Xia, Q. (2005). A hybrid algorithm for a class of resource-constrained scheduling problems. In R. Bartak & M. Milano (Eds.). *CPAIOR'2005, Lecture Notes in Computer Science 3524*, (110-124). Germany: Springer-Verlag.

Daniels, R.L., Hoopes, B.J., & Mazzola, J.B. (1996). Scheduling parallel manufacturing cells with resource flexibility. *Management Science*, *42* **(**9), 1260-1276.

Daniels, R. L., Hoopes, B. J., & Mazzola, J. B. (1997). An analysis of heuristics for the parallel-machine flexible-resource scheduling problem. *Annals of Operations Research*, *70*, 439-472.

Daniels, R.L., Hua, S.Y., & Webster, S. (1999). Heuristics for parallel-machine flexible-resource scheduling problems with unspecified job assignment. *Computers & Operations Research*, *26*, 143-155.

Dantzig, G.B. (1963). *Linear programming and extensions*, New Jersey, Princeton: Princeton University Press.

Darbi-Dowman, K. D., & Little, J. (1998). Properties of some combinatorial optimization problems and their effect on the performance of integer programming and constraint logic programming. *Informs Journal on Computing*, *10*(3), 276-286.

Darbi-Dowman, K.D., Little, J., Mitra, G., & Zaffalon, M. (1997) Constraint logic programming and integer programming approaches and their collaboration in solving an assignment scheduling problem. *Constraints*, *1*, 245-264.

Edis, E.B., Araz, C., & Ozkarahan, I. (2008). Lagrangian-based solution approaches for a resource-constrained parallel machine scheduling problem with machine eligibility restrictions. In N.T. Nguyen et al., (Eds.). *IEA/AIE 2008, Lecture Notes in Artificial Intelligence 5027* (337–346). Germany: Springer-Verlag

Edis, E.B., Mizrak Ozfirat, P., & Ozkarahan, I. (2008). A constraint programming approach for a conference timetabling problem. *Proceedings of the 37th Annual Meeting of the Western Decision Sciences Institute*, 603-605.

Edis, E.B., & Ozkarahan, I. (2006). Comparison of integer programming, constraint programming and hybrid approaches for a class of scheduling problems.

Proceedings of the 26th National Conference on Operations Research and Industrial Engineering, 281-284.

Edis, E.B., & Ozkarahan, I. (2007). Parallel machine scheduling with additional resources: A brief review of the literature. *Proceedings of 11th IFAC International Symposium CEFIS'2007*, 381-386.

Edis, E.B., & Ozkarahan, I. (2008). Parallel machine scheduling problem with additional resources and machine eligibility restrictions: An iterative integer programming approach. *International Journal of Computers, Information Technology and Engineering*, *2*(1), 1-11.

Eiselt, H.A., & Sandblom, C-L. (2004). *Decision analysis, location models, and scheduling problems*. New York: Springer-Verlag Berlin Heidelberg.

Eiselt, H.A., & Sandblom, C-L. (2007). *Linear Programming and its Applications*. New York: Springer-Verlag Berlin Heidelberg.

Fisher, M. L. (1973). Optimal solution of scheduling problems using Lagrange multipliers: Part I. *Operations Research*, *21*, 1114-1127.

Fisher, M. L. (1981). The Lagrangian relaxation method for solving integer programming problems. *Management Science*, *27*(1), 1-18.

Focacci, F., Lodi, A., & Milano, M. (2002). Mathematical programming techniques in constraint programming: A short overview. *Journal of Heuristics*, *8*, 7-17.

Frangioni, A. (2005). About Lagrangian methods in integer optimization. *Annals of Operations Research*, *139*, 163–193.

Fumero, F. (2001). A modified subgradient algorithm for Lagrangian relaxation. *Computers and Operations Research 28*(1), 33–52.

Gao, L., Wang, C., Wang, D., Yin, Z., & Wang, S. (1998). A production scheduling system for parallel machines in an electrical appliance plant. *Computers and Industrial Engineering*, *35*(1-2), 105-108.

Garey, M.R., & Graham, R.L. (1975). Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, *4*(2), 187-200.

Garey, R. L., Graham, D. S., & Johnson, A. (1976). Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory, Series A*, *21*(3), 257-298.

Garey, M.R., & Johnson, D.S. (1975). Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, *4*(4), 397-411.

Garey, M.R., & Johnson, D.S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco: W. H. Freeman and Company.

Geoffrion, A.M. (1974). Lagrangian Relaxation for integer programming. *Mathematical Programming Study*, *2*, 82-114.

Glass, C.A., & Kellerer, H. (2007). Parallel machine scheduling with job assignment restrictions. *Naval Research Logistics*, *54*(*3*), 250–257.

Glass, C. A., Shafransky, Y.M., & Strusevich, V.A. (2000). Scheduling for parallel dedicated machines with a single server. *Naval Research Logistics*, *47*(4), 304-328.

Goffin, J.L. (1977). On the convergence rates of subgradient optimization methods. *Mathematical Programming*, *13*, 329–347.

Gomory, R. (1963). Recent advances in mathematical programming. In R. Graves & P. Wolfe (Eds.). *An algorithm for integer solutions to linear programs* (260-302). New York: McGraw-Hill.

Gonzalez, M.C. (1977). Deterministic Processor Scheduling, *ACM Survey*, *9*, 173-204.

Graham, R.L., Lawler, E.L., Lenstra, J.K., & Rinnooy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, *5*, 287-326.

Grigoriev, A., Sviridenko, M., & Uetz, M. (2005). Unrelated parallel machine scheduling with resource dependent processing times. *Mathematical Programming Series B*, *110*, 209-228.

Grigoriev, A., Sviridenko, M., & Uetz, M. (2006). LP rounding and an almost harmonic algorithm for scheduling with reource dependent processing times. In J. Diaz et al. (Eds.). *APPROX and RANDOM 2006, Lecture Notes in Computer Science 4110* (140-151). Germany: Springer-Verlag Berlin Heidelberg.

Grigoriev, A., Sviridenko, M., & Uetz, M. (2007). Machine scheduling with resource dependent processing times. In M. Jünger & V. Kaibel (Eds.). *IPCO 2005, Lecture Notes in Computer Science 3509* (182-195). Germany: Springer-Verlag.

Grigoriev, A., & Uetz, M. (2006). Scheduling parallel jobs with linear speedup. In T. Erlebach & G. Parsiano (Eds.). *WAOA 2005, Lecture Notes in Computer Science 3879* (203-215). Germany: Springer-Verlag Berlin Heidelberg

Grigoriev, A., & Uetz, M. (2009). Scheduling jobs with time-resource trade-off via non-linear programming. *Discrete Optimization*, *6*, 414-419.

Guignard, M. (2002). Lagrangian relaxation. In P.M. Pardalos & G.C.R. Mauricio (Eds.). *Handbook of Applied Optimization* (465-474). New York: Oxford University Press.

Guignard, M. (2003). Lagrangean relaxation. *TOP*, *11*(2), 151–228.

Hall, N.G., Potts, C. N., & Sriskandarajah, C. (2000). Parallel machine scheduling with a common server. *Discrete Applied Mathematics*, *102*, 223-243.

Harvey, N.J.A., Ladner, R.E., Lovasz, L., & Tamir, T. (2006). Semi-matchings for bipartite graphs and load balancing. *Journal of Algorithms*, *59*(1), 53–78.

Heipcke, S. (1999). Comparing constraint programming and mathematical programming approaches to discrete optimisation-the change problem. *Annals of Operations Research*, *50*, 581-595.

Held, M., & Karp, R.M. (1970). The traveling salesman problem and minimum spanning trees. *Operations Research*, *18*, 1138-1162.

Held, M., & Karp, R.M. (1971). The traveling salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, *1*, 6-25.

Held, M., Wolfe P., & Crowder, H. (1974). Validation of subgradient optimization. *Mathematical Programming*, *6*, 62-88.

Hooker, J. (2000). *Logic-based methods for optimization: combining optimization and constraint satisfaction*. USA: Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons.

Hooker, J.N. (2002). Logic, optimization and constraint programming. *INFORMS Journal on Computing*, *14*, 295 - 321.

Hooker, J.N. (2005). A hybrid method for planning and scheduling. *Constraints*, *10*, 385-401.

Hooker, J. N. (2006). An integrated method for planning and scheduling to minimize tardiness. *Constraints*, *11***,** 139-157.

Hooker, J.N. (2007). *Integrated Methods for Optimization*. USA: Springer Science+Business Media.

ILOG, 2003. *OPL Studio 3.7 Language manual*. France: ILOG S.A.

ILOG, 2005a. *CPLEX 9.0 User's Manual*. France: ILOG S.A.

ILOG, 2005b. *ILOG Solver 6.0 User's Manual*. France: ILOG S.A.

ILOG, 2005c. *ILOG Scheduler 6.0 User's Manual*. France: ILOG S.A.

Jain, V., & Grossmann, I.E. (2001). Algorithms for hybrid MILP/CP models for a class of optimization problems. *Informs Journal on Computing*, *13*, 258-276.

Ji, M., & Cheng, T.C.E. (2008). An FPTAS for parallel-machine scheduling under a grade of service provision to minimize makespan. *Information Processing Letters*, *108*(4), 171–174.

Jozefowska, J., & Weglarz, J. (2004). Scheduling with resource constraints-continuous resources. In J.Y-T. Leung (Ed.). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (Chapter 24). USA: CRC Press.

Kanet, J. J., Ahire, S. L., & Gorman, M. F. (2004). Constraint programming for scheduling. In J.Y-T. Leung (Ed.). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (Chapter 47). USA: CRC Press.

Kedad-Sidhoum, S., Solis, Y.R., & Sourd, F. (2008). Lower bounds for the earliness-tardiness scheduling problem on single and parallel machines. *European Journal of Operations Research*, *189*, 1305-1316.

Kelbel, P, & Hanzalek, Z. (in press). Solving production scheduling with earliness/tardiness penalties by constraint programming. *Journal of Intelligent Manufacturing*. doi: 10.1007/s10845-009-0318-2.

Kellerer, H. (2008). An approximation algorithm for identical parallel machine scheduling with resource dependent processing times. *Operations Research Letters*, *36*, 157-159.

Kellerer, H., & Strusevisch, V.A. (2003). Scheduling parallel dedicated machines under a single non-shared resource. *European Journal of Operational Research*, *147*, 345-364.

Kellerer, H., & Strusevisch, V.A. (2004). Scheduling problems for parallel dedicated machines under multiple resource constraints. *Discrete Applied Mathematics, 133*, 45-68.

Kellerer, H., & Strusevisch, V.A. (2008). Scheduling parallel dedicated machines with the speeding-up resource. *Naval Research Logistics, 55*(5), 377-389.

Kovalyov, M.Y., & Shafransky, Y. M. (1998). Uniform machine scheduling of unit-time jobs subject to resource constraints. *Discrete Applied Mathematics*, *84*, 253-257.

Krause, K.L., Shen, V.Y., & Schwetman, H. D. (1973). A task scheduling algorithm for a multiprogramming computer system. *ACM SIGOPS Operating Systems*, *7* (4), 112-118.

Krause, K.L., Shen, V.Y., & Schwetman, H. D. (1975). Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *Journal of the Association for Computing Machinery*, *22*(4), 522-550.

Lemarechal, C. (2001). Lagrangian relaxation. In M. Junger & D. Naddef (Eds.). *Computational Combinatorial Optimization, Lecture Notes in Computer Science 2241* (115-160). Germany: Springer-Verlag Heidelberg.

Lenstra, J.K., Shmoys, D.B., & Tardos, E. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, *46*(1–3), 259–271.

Leung, J.Y-T, & Li, C-H. (2008). Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, *116*, 251-262.

Li, C.-L. (2006). Scheduling unit-length jobs with machine eligibility restrictions. *European Journal of Operational Research*, *174*(2), 1325–1328.

Li, C.-L., & Wang, X. (2009). Scheduling parallel machines with inclusive processing set restrictions and job release times. *European Journal of Operational Research*, doi:10.1016/j.ejor.2009.02.011, In Press.

Li, Y., Wang, F., & Lim A. (2003). Resource constraints machine scheduling: A genetic algorithm approach, *CEC: 2003 Congress on Evolutionary Computation*, *1-4*, 1080-1085.

Lim M-F., & Karimi I.A. (2003). Resource-constrained scheduling of parallel production lines using asynchronous slots. *Industrial & Engineering Chemistry Research*, *42*(26), 6832-6842.

Lin, Y., & Li, W. (2004). Parallel machine scheduling of machine-dependent jobs with unit-length. *European Journal of Operational Research*, *156*(1), 261–266.

Lin, C.K.Y., Wong, C.L., & Yeung, Y.C. (2002). Heuristic approaches for a scheduling problem in the plastic molding department of an audio company. *Journal of Heuristics*, *8*, 515-540.

Lorena, L.A.N., & Senne, E.L.F. (1999). Improving traditional subgradient scheme for Lagrangean relaxation: an application to location problems. *International Journal of Mathematical Algorithms*, *1*, 133–151.

Luh, P.B., & Hoitomt, D.J. (1993). Scheduling of manufacturing systems using the Lagrangian relaxation technique. *IEEE Transactions on Robotics and Automation*, *38*(7), 1066-1079.

Luh, P.B., Hoitomt, D.J., Max, E., & Pattipati, K.R. (1990). Schedule generation and reconfiguration for parallel machines. *IEEE Transactions on Robotics and Automation*, *6*(6), 687-696.

Lustig I. J., & Puget J. F. (2001). Program does not equal program: Constraint programming and its relationship to mathematical programming. *Interfaces*, *31*(6), 29-53.

Magatao, L. (2005). *Mixed integer linear programming and constraint logic programming: A unified modeling framework*. PhD Thesis, Graduate School in Electrical Engineering and Industrial Computer Science, Brasil.

Martello, S., Soumis, F., & Toth, P. (1997). Exact and approximation algorithms for makespan minimization on unrelated parallel machines. *Discrete Applied Mathematics*, *75*, 169-188.

McNaughton, R. (1959). Scheduling with deadlines and loss functions. *Management Science*, *6*(1), 1-12.

Mendez, C.A., & Cerda, J. (2004). An MILP framework for batch reactive scheduling with limited discrete resources. *Computers & Chemical Engineering*, *28*(6-7), 1059-1068.

Mendez, C.A., Cerda, J., Grossmann, I.E., Harjunkoski, I., & Fahl, M. (2006). State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & Chemical Engineering*, *30*(6-7), 913-946.

Milano, M. (Ed). (2004). *Constraint and integer programming: Toward a unified methodology*. USA: Kluwer Academic Publishers.

Milano, M., Ottosson, G., Refalo, P., & Thorsteinsson, E. S. (2002). The role of integer programming techniques in constraint programming's global constraints. *Informs Journal on Computing, 14*(4), 387-402.

Milano, M., & Trick, M. (2004). Constraint and integer programming. In M. Milano (Ed). *Constraint and integer programming: Toward a unified methodology* (1-33). USA: Kluwer Academic Publishers.

Milano, M., & Wallace, M. (2006). Integrating operations research in constraint programming. *4OR*, *4*, 175-219.

Mızrak Ozfirat, P., Edis, E.B., & Ozkarahan, I. (2006). Mathematical modeling approach for the course timetabling problem of Dokuz Eylul University Industrial Engineering Department. *Proceedings of the International Conference on Modeling and Simulation*, 853-857.

Mokotoff, E. (2001). Parallel Machine Scheduling Problems: A Survey. *Asia Pacific Journal of Operational Research*, *18*, 193-242.

Nagarur, N., Vrat, P., & Duongsuwan, W. (1997). Production planning and scheduling for injection molding of pipe fittings. *International Journal of Production Economics*, *53*, 157-170.

Nagendra, P., Das, S.K., & Nathan, S. (2000). Deriving the detailed machine schedule from a weekly MRP requirement. *Production Planning and Control*, *11*(6), 547-555.

Olafsson, S., & Shi, L. (2000). A method for scheduling in parallel manufacturing systems with flexible resources. *IIE Transactions*, *32*, 135-146.

Ou, J., Leung, J.Y-T., & Li, C-L. (2008). Scheduling parallel machines with inclusive processing set restrictions. *Naval Research Logistics*, *55*(4), 328–338.

Ozdamar, L., & Ulusoy, G. (1994). A local constraint based analysis approach to project scheduling under general resource constraints. *European Journal of Operational Research*, *79*, 287-298.

Ozkarahan, I., Edis, E. B., & Mizrak Ozfirat, P. (2009). Scheduling of surgical operations to operating rooms by mathematical modeling. *Proceedings of Decision Sciences Institute, 40th Annual Meeting Conference*, 921-926.

Pfund, M., Fowler, J. W., & Gupta, J. N. D. (2004). Multi-objective unrelated parallel-machine deterministic scheduling problems. *Journal of the Chinese Institute of Industrial Engineers, 21*(3), 230-241.

Pinedo, M. (1995). *Scheduling: Theory, algorithms and systems*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Pinedo, M. (2008). *Scheduling: Theory, algorithms and systems* (3rd ed.). USA: Springer Science+Business Media.

Pinedo, M., & Chao, X. (1999). *Operations scheduling with applications in manufacturing and services*. Singapore: McGraw-Hill.

Pinto, J.M., & Grossman, I.E. (1997). A logic-based approach to scheduling problems with resource constraints. *Computers and Chemical Engineering*, *21*(8), 801-818.

Reeves, C.R. (Ed.). (1995). *Modern heuristic techniques for combinatorial problems*. UK: McGraw-Hill.

Rodosek, R., Wallace, M.G., & Hajian, M.T. (1999). A new approach to integrating mixed integer programming and constraint logic programming. *Annals of Operations Research*, *86*, 63-87.

Ruiz-Torres, A.J. & Centeno, G. (2007). Scheduling with flexible resources in parallel workcenters to minimize maximum completion time. *Computers & Operations Research, 34*, 48-69.

Ruiz-Torres A.J., Lopez, F.J., & Ho, J.C. (2007). Scheduling uniform parallel machines subject to a secondary resource to minimize the number of tardy jobs. *European Journal of Operational Research*, *179*(2), 302-315.

Shabtay, D., & Kaspi, M. (2006). Parallel machine scheduling with a convex resource consumption function. *European Journal of Operational Research*, *173*, 92-107.

Shchepin, E.V., & Vakhania, N. (2005). An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, *33*(2), 127–133.

Slowinski, R. (1980). Two approaches to problems of resource allocation among project activities – A comparative study. *Journal of the Operational Research Society*, *31*(8), 711-723.

Smith, B. (1995). *A tutorial on constraint programming*. Research Report 95.14, School of Computer Studies, University of Leeds.

Smith, B.M., Brailsford, S.C., Hubbard, P.M., & Williams, H.P. (1997). The progressive party problem: integer linear programming and constraint programming compared. *Constraints*, *1*, 119-138.

Srivastav, A., & Stangier, P. (1997). Tight approximations for resource constrained scheduling and bin packing. *Discrete Applied Mathematics*, *79*, 223-245.

Sue, L-H., & Lien, C-Y. (2009). Scheduling parallel machines with resource-dependent processing times. *International Journal of Production Economics*, *117*, 256-266.

Tamaki, H., Hasegawa, Y., Kozasa, J., & Araki, M. (1993). Application of search methods to scheduling problem in plastics forming plant: a binary representation approach. *Proceedings of the 32nd IEEE Conference on Decision and Control*, 3845-3850.

Thorsteinsson, E.S. (2001a). Branch-and-Check: A hybrid framework integrating mixed integer programming and constraint logic programming. In T. Walsh (Ed.). *Principles and practice of constraint programming*, *Lecture Notes in Computer Science 2239*, (16-30), Germany: Springer-Verlag Berlin Heidelberg.

Thorsteinsson, E.S. (2001b). *Hybrid approaches to combinatorial optimisation*. PhD Thesis, Carnegie Mellon University, Graduate School of Industrial Administration, USA.

Topaloglu, S., & Ozkarahan, I. (2004). Comparison of different variable and value order strategies for the optimum solution of a single machine scheduling problem with sequence-dependent setups. In C. Aykanat, T. Dayar & I. Körpeoglu (Eds.). *ISCIS 2004, Lecture Notes in Computer Science, 3280,* (996-1005). Germany: Springer-Verlag.

Vairaktarakis, G.L., & Cai, X. (2003). The value of processing flexibility in multipurpose machines. *IIE Transactions*, *35*, 763-774.

Van den Akker, J. M, Hurkens, C.A.J., & Savelsbergh, M.W.P. (2000). Time-indexed formulations for machine scheduling problems: Column generation. *Informs Journal on Computing*, *12*(2), 111-124.

Van Hentenryck, P. (1999). *The OPL™ Optimization Programming Language*. Cambridge MA.: MIT Press.

Van Hentenryck, P., Perron, L., & Puget J-F. (2000). Search and strategies in OPL. *ACM Transactions on Computational Logic (TOCL)*, *1*(2), 285-320.

Ventura, J.A., & Kim, D. (2000). Parallel machine scheduling about an unrestricted due date and additional resource constraints. *IIE Transactions*, *32*, 147-153.

Ventura, J.A., & Kim, D. (2003). Parallel machine scheduling with earliness-tardiness penalties and additional resource constraints. *Computers and Operations Research, 30,* 1945-1958.

Wallace, M.G. (2007). Hybrid algorithms in constraint programming. In F. Azevedo et al. (Eds.). *CSCLP 2006, Lecture Notes in Artificial Intelligence, 4651*, 1–32, Springer-Verlag Berlin Heidelberg.

Wallace, M., Novello, S., & Schimpf, J. (1997). ECLIPSe: A platform for constraint logic programming. *ICL Systems Journal*, *12*, 159-200.

Wang, S-H. (2003). An improved stepsize of the subgradient algorithm for solving the Lagrangian relaxation problem. *Computers and Electrical Engineering, 29*, 245-249.

Williams, H. P. (1999). *Model building in mathematical programming* (4th ed.). England: John Wiley & Sons.

Yu, L., Shih, H.M., Pfund, M., Carlyle, W.M., & Fowler, J.W. (2002). Scheduling of unrelated parallel machines: An application to PWB manufacturing, *IIE Transactions*, *34*, 921-931.

APPENDICES

APPENDIX A. Summary of surveyed papers

Studies	Problem Size in Computational Exp.	Additional Resource Characteristics	Machine Environment	Objective	Solution Method(s)	Other Important Issues
Blazewicz (1979)	-	One additional resource type with arbitrary size and 0/1 resource requirements	Identical machines	-	A polynomial-time algorithm for $P\|res1\cdot 1, r_i, d_i\|\varnothing$	-
Blazewicz, Lenstra & Rinnooy Kan *et al.* (1983)	-	Various versions of the problem	- Identical machines - uniform machines	makespan	O($n$log$n$) algorithm for $Q2\|res1\cdot\cdot, p_i=1\|C_{\max}$	They prove that below cases are NP-hard. - $P3\|res\cdot 11, p_i=1\|C_{\max}$ - $Q2\|res\cdot 11, p_i=1\|C_{\max}$
Blazewicz and Ecker (1983)	-	-Fixed number of additional resource types, - Fixed resource size - A fixed upper bound on resource requirements	Identical machines	makespan	O(log$n$) algorithm for $P\|res\lambda\sigma\delta, p_i=1\| C_{max}$	-
Blazewicz , Barcelo, Kubiak, & Röck (1986)	-	-One additional resource type with arbitrary size and arbitrary resource requirements -Arbitrary number of additional resource types with one unit and 0/1 resource requirements	Identical parallel two machines	lateness	Proves that $P2\|res1\cdot\cdot, p_i=1\|L_{\max}$ and $P2\|res\cdot\cdot 1, p_i=1 \mid L_{\max}$ problems are NP-hard.	Proves that $P2\|res1\cdot\cdot, p_i=1\|L_{\max}$ problem is reducible to $P2\|res\cdot\cdot 1, p_i=1 \mid L_{\max}$
Blazewicz, Kubiak, Röck & Szwarcfiter (1987).	-	-One additional resource type with arbitrary size and 0/1 resource requirements -One additional resource type with arbitrary size and arbitrary resource requirements -Arbitrary number of additional resource types with one unit and 0/1 resource requirements	Identical machines	flow time	- Proves that $P\|res1\cdot 1\|\sum_i C_i$ can be solved in O(n^3) time -Proves that $P2\|res1\cdot\cdot\|\sum_i C_i$ and $P2\|res\cdot\cdot 1\|\ \sum_i C_i$ problems are NP-hard.	-
Blazewicz , Kubiak & Martello (1993)	- 10, 20, 50, 100, 250, 500, 1000 jobs - Two (fixed) machines - 1, 2, 5, 10 additional resource types - Resource sizes (R_l) in [1,100] - Resource req. in [0,R_l]	-Arbitrary number of additional resource types with arbitrary size and arbitrary resource requirements	Identical machines	lateness	- Problem-based heuristic algorithms - Taking the solution of best heuristic algorithm as initial feasible upper bound, a B&B algorithm	- Various lower bounds are proposed to be used in B&B algorithm.

APPENDIX A. Summary of surveyed papers (Continuing)

Studies	Problem Size in Computational Exp.	Additional Resource Characteristics	Machine Environment	Objective	Solution Method(s)	Other Important Issues
Daniels, Hoopes & Mazzola (1996)	- 10, 15 jobs and 2, 3, 4 machines, - 20, 50, 80 jobs and 2, 3, 4 machines.	- A PMFRS problem; $R = m$, m+1, m+2 - Static and dynamic resource allocations.	Dedicated manufacturing cells	makespan	For the *static* PMFRS problem: - A polynomial time exact algorithm For the *dynamic* PMFRS problem: - A B&B algorithm - A static-based heuristic algorithm - A B&B initialized static-based heuristic.	- First used the term RCPMSP
Daniels, Hoopes & Mazzola (1997)	- 10, 15 jobs and 2, 3, 4 machines, - 20, 50, 80 jobs and 2, 3, 4 machines.	- A PMFRS problem, $R = m$, m+1, m+2 - Static and dynamic resource allocations.	Dedicated manufacturing cells	makespan	For the *dynamic* PMFRS problem - A B&B algorithm - A tabu search heuristic - A static-based tabu search heuristic	-
Daniels, Hua and Webster (1999)	- 10 and 15 jobs and 3, 4 machines - 20, 30, 50 jobs and 3, 4, 5 machines	- An UPMFRS problem, $R = m$, m+1, m+2 - Static resource allocation.	Identical machines	makespan	- A decomposition heuristic - A tabu search heuristic	-
Garey and Johnson (1975)	-	One and arbitrary number of additional resource types with arbitrary size and requirements	Identical machines	makespan	A polynomial-time algorithm for $P2\|res\cdots, p_i=1\|C_{\max}$	Prove that the problem of $P3\|res1\cdots, r_i, p_i=1\|C_{\max}$ is NP-hard.
Grigoriev and Uetz (2006)	-	A PMFRS problem	Dedicated machines	makespan	- A relaxed quadratic-programming based $(3+\varepsilon)$-approximation scheme (greedy algorithm)	- Relaxed formulation gives a lower bound and assignment of resources to jobs.
Grigoriev and Uetz (2009)	-	A PMFRS problem	Dedicated machines	makespan	- Utilizing a (nonlinear) integer program, a polynomial time $(3+\varepsilon)$-approximation algorithm.	- The authors derived lower bounds for the approximation. - The relaxed nonlinear program gives assignment of resources to jobs
Grigoriev, Sviridenko & Uetz (2005)	-	- A PMFRS problem - A UPMFRS problem	-Unrelated machines -Dedicated machines	makespan	Based on a rounding algorithm: - $(4+2\sqrt{2})$-approximation (LP-Greedy) algorithm for unrelated machine case - $(3+2\sqrt{2})$-approximation (LP-Greedy) algorithm for dedicated machines case	- A two-phase rounding algorithm for the relaxed formulation of the problem.
Grigoriev Sviridenko & Uetz (2006)	-	A PMFRS problem	Unrelated machines	makespan	Based on a LP-rounding algorithm an 3.75-approximation algorithm	- The relaxed formulation of the problem gives assignment of resources to jobs.

APPENDIX A. Summary of surveyed papers (Continuing)

Studies	Problem Size in Computational Exp.	Additional Resource Characteristics	Machine Environment	Objective	Solution Method(s)	Other Important Issues
Grigoriev Sviridenko & Uetz (2007)	-	A PMFRS problem	Unrelated machines	makespan	Based on a LP rounding algorithm - 4-approximation LP-Greedy algorithm - 3.75-approximation (improved LP-Greedy) algorithm	- An LP rounding algorithm for the relaxed formulation gives assignment of resources to jobs.
Kellerer (2008)	-	A PMFRS problem	Identical machines	makespan	$(3.5+\varepsilon)$- approximation algorithm	- Utilize the aggregate version of resource constraints.
Kellerer and Strusevisch (2003)	-	- One additional resource type - Resource requirements of jobs are either zero or one. - Unit Resource size $PDm\|res111\|C_{\max}$	Dedicated machines	makespan	-Polynomial time exact algorithm for $PD2\|res111\|C_{\max}$ - Polynomial time approx. algorithms for $PD3\|res111\|C_{\max}$ $PD4\|res111\|C_{\max}$ $PDm\|res111\|C_{\max}$	- Proves that $PD3\|res111\|C_{\max}$ $PDm\|res111\|C_{\max}$ problems are NP-hard.
Kellerer and Strusevisch (2004)	-	Various versions of the problem	Dedicated machines	makespan	- Polynomial time algorithms for $PD2\|res1\cdot\cdot\|C_{\max}$, $PD2\|res211\|C_{\max}$ -A simple greedy approximation algorithm for $PD\|res\lambda 11\|C_{\max}$ - A PTAS for $PDm\|res\lambda 11\|C_{\max}$	- Proves that $PD2\|res222\|C_{\max}$ $PD2\|res311\|C_{\max}$ and $PD3\|res211\|C_{\max}$ problems are NP-hard.
Kellerer and Strusevisch (2008)	-	- The case of one additional resource with the unit size and are 0/1 resource requirements. - The case of one additional resource with arbitrary size and arbitrary resource requirements of jobs	Dedicated machines	makespan	- Polynomial time algorithm for $PD2\|res111, Bi\|C_{\max}$ - (3/2)-approximation algorithm for $PD\|res111, Bi\|C_{\max}$ - 3-approximation algorithm for $PD\|res1\sigma\sigma, Int\|C_{\max}$ - A polynomial time approximation algorithm for $PDm\|res111, Bi\|C_{\max}$	-All approximation algorithms are established on knapsack sub-problems -Define *Bi*, *Lin*, *Int* for the β field of $\alpha\|\beta\|\gamma$ scheme
Kovalyov and Shafransky (1998)	-	- One additional resource type. - Resource sizes are arbitrary. - Resource requirements are 0/1 $Q\|res1\cdot 1, p_i=1\|C_{\max}$ problem	Uniform parallel machines	makespan	-A O(m log m) algorithm for $Q\|res1\cdot 1, p_i=1, nmit\|C_{\max}$ -A linear time algorithm for $P\|res1\cdot 1, p_i=1\|C_{\max}$	- The no machine idle time (*nmit*) case is investigated. - Unit processing times, $p_i = p$
Krause, Shen & Schwetman (1975)	-	-One additional resource type (memory) with arbitrary size - Arbitrary memory requirements of jobs	Identical processors	makespan	- Heuristic scheduling strategies for unit and arbitrary processing time cases.	- Worst case bound analyzes of various algorithms for both unit and arbitrary processing time cases.

APPENDIX A. Summary of surveyed papers (Continuing)

Studies	Problem Size in Computational Exp.	Additional Resource Characteristics	Machine Environment	Objective	Solution Method(s)	Other Important Issues
Li, Wang & Lim (2003)	- 3 to 40 machines - 10 jobs per machine	- One additional resource with arbitrary sizes and requirements	Dedicated machines	makespan	Genetic algorithm	They propose three kinds of lower bounds
Olafsson and Shi (2000)	- 10,15, 20 jobs - 3, 4, 5 machines	- A PMFRS problem; $R = m$, m+1, m+2	Dedicated manufacturing cells	makespan	- A heuristic method called Nested Partitions (NP) (an adaptive sampling method that combines local and global search)	Dynamic resource allocation is compared with static one
Ruiz-Torres and Centeno (2007)	- Medium size: 20, 30, 50 jobs; 3, 4, 5 machines and - Large size: 50, 100 jobs; 5, 10 machines	- A PMFRS problem $R = m$+1, m+2 for medium size $R = 2m$, 3m, $4m$ for large size - Static Resource Allocation	Identical machines	makespan	- Problem-based heuristic algorithms and comparison with the algorithms of Daniels *et al.* (1999)	- Provides a method to estimate the lower bound - Pooling resources into fewer work centers improves C_{max}
Ruiz-Torres, Lopes and Ho (2007)	- Small instances with m = 2, n/m=5 and 7 - Large instances with m = 5, 10; n/m=10, 20 - 50 experiments for each point.	- A PMFRS problem - For all experiments R/m = 3, 6 - Static Resource Allocation	Uniform Machines	number of tardy jobs	- Five different problem-based heuristic algorithms and a detailed analyze of these heuristics at all experimental points.	- The heuristics that allocate jobs to machines one machine at a time outperform the ones with simultaneous allocation of machines.
Srivastav and Stangier (1997)	-	- Arbitrary number of additional resource types with arbitrary sizes but only 0/1 resource requirements	Identical processors	makespan	- A polynomial time approximation algorithm for $P\|res\cdot\cdot1, r_i, p_i=1 \mid C_{max}$	- p_i=1 (unit time jobs) - release time of jobs (r_i) are also considered.
Sue and Lien (2009)	-3,5,8,10 machines -3m+1, 3m+2, 4m+1, 4m+2, 5m+1, 5m+2 jobs	- A PMFRS problem $R = n$, 3n/2, 2n	Identical machines	makespan	- Various problem based heuristic algorithms	- Deploying resources optimally in advance and then assigning the jobs effectively yields very favorable schedules.
Ventura and Kim (2000)	A small example with five jobs and two machines	- One additional resource type - Resource sizes are arbitrary - Resource requirements are 0/1	Identical machines	*TAD*	It is proved that $P\|res1\cdot1, p_i, d_i=d\|$ *TADD* problem is solved in polynomial time.	Dominance properties are given
Ventura and Kim (2003)	- 100, 200, 300 jobs - 2, 3, 5 machines	- One, two, three resource types - (6,10); (9,15); (15,25) resource sizes for two, three and five machines respectively - Resource requirements are arbitrary within interval (0,5)	Identical machines	*TADD*	- Lagrangian Relaxation-based Lagrangian Heuristic	- Due dates (20, 30); (30,60); (30,80) - Ready times (0,10) - Unit (same) processing times of jobs , $p_i = p$

APPENDIX B. Computational results (LSA and PSH)

					Lagrangian-based Solution Approach (LSA)						Problem Specific Heuristic (PSH)	
					IH	LH						
n	*m*	*b*	F_p	Sample No		Lagr. LB	LH	# of Iter	Time (sec.)	% (UB-LB)/LB	PSH	% (UB-LB)/LB
50	3	1	Low	1	561	481.963	482	125	17.01	0.00%	483	0.21%
				2	675	521.765	526	193	28.80	0.77%	522	0.00%
				3	688	555.017	556	46	6.53	0.00%	556	0.00%
				4	796	617.517	619	202	36.95	0.16%	634	2.59%
				5	540	481.984	482	127	15.81	0.00%	482	0.00%
				6	583	481.464	482	59	8.47	0.00%	502	4.15%
				7	617	537.141	538	57	8.39	0.00%	538	0.00%
				8	654	555.03	556	76	10.40	0.00%	576	3.60%
				9	523	455.643	456	120	15.32	0.00%	504	10.53%
				10	608	500.4	508	160	21.64	1.40%	522	4.19%
				11	785	575.751	579	194	28.30	0.52%	595	3.30%
				12	706	574.081	575	49	7.12	0.00%	581	1.04%
				13	562	476.972	482	163	21.57	1.05%	524	9.85%
				14	524	467.121	479	166	20.02	2.35%	472	0.85%
				15	663	522.498	527	183	22.88	0.76%	534	2.10%
				16	622	514.422	522	180	27.76	1.36%	528	2.52%
				17	613	500.348	508	168	25.74	1.40%	507	1.20%
				18	618	500.337	507	179	26.55	1.20%	508	1.40%
				19	610	493.692	494	75	10.05	0.00%	505	2.23%
				20	651	529.67	538	179	27.65	1.51%	538	1.51%
				21	614	514.323	522	161	24.39	1.36%	527	2.33%
				22	667	546.923	556	162	26.90	1.65%	556	1.65%
				23	670	546.855	556	179	29.25	1.65%	564	3.11%
				24	638	529.99	538	183	26.97	1.51%	540	1.89%
				25	774	629.511	642	174	34.88	1.90%	642	1.90%
				26	615	500.497	514	183	25.29	2.59%	523	4.39%
				27	685	537.707	544	179	26.41	1.12%	539	0.19%
				28	570	487.971	494	182	26.91	1.23%	494	1.23%
				29	548	479.434	494	185	23.66	2.92%	534	11.25%
				30	674	557.909	562	172	24.22	0.72%	568	1.79%
Average					-	-	-	148.70	21.86	0.97%	-	2.70%
50	3	1	High	1	561	481.932	486	180	30.38	0.83%	482	0.00%
				2	673	555.994	570	174	37.57	2.52%	556	0.00%
				3	524	467.142	478	160	26.35	2.14%	472	0.85%
				4	594	493.987	494	140	26.67	0.00%	494	0.00%
				5	692	585.246	596	180	42.31	1.71%	596	1.71%
				6	650	529.823	539	172	37.57	1.70%	538	1.51%
				7	711	565.08	591	165	36.12	4.42%	575	1.59%
				8	630	521.966	534	179	36.43	2.30%	522	0.00%
				9	622	514.259	532	168	33.07	3.30%	522	1.36%
				10	571	555.99	570	183	40.06	2.52%	556	0.00%
				11	669	546.618	568	183	38.75	3.84%	572	4.57%
				12	600	514.471	522	180	34.99	1.36%	522	1.36%
				13	552	476.944	482	162	28.13	1.05%	482	1.05%
				14	808	653.992	669	182	54.75	2.29%	667	1.99%
				15	760	606.491	635	168	43.08	4.61%	618	1.81%
				16	613	506.995	520	176	35.17	2.56%	507	0.00%
				17	595	500.473	507	177	33.63	1.20%	507	1.20%
				18	666	546.913	557	180	39.68	1.83%	556	1.65%
				19	695	537.993	552	170	35.35	2.60%	540	0.37%
				20	601	488.485	494	189	34.86	1.02%	494	1.02%
				21	561	482	490	166	26.53	1.66%	482	0.00%
				22	638	529.815	550	168	34.77	3.77%	538	1.51%
				23	546	476.569	482	169	30.53	1.05%	482	1.05%
				24	618	521.954	522	126	25.47	0.00%	522	0.00%
				25	606	521.708	522	84	16.24	0.00%	522	0.00%
				26	557	476.907	482	169	29.04	1.05%	482	1.05%
				27	532	462.88	468	179	28.31	1.08%	477	3.02%
				28	507	462.468	469	166	28.06	1.30%	463	0.00%
				29	645	537.999	541	172	38.83	0.56%	538	0.00%
				30	622	514.483	531	169	30.34	3.11%	522	1.36%
Average					-	-	-	167.8	33.77	1.91%	-	1.00%

APPENDIX B. Computational results (LSA and PSH) - *Continues*

					Lagrangian-based Solution Approach (LSA)						Problem Specific Heuristic (PSH)	
					IH	LH						
n	*m*	*b*	F_p	Sample No		Lagr. LB	LH	# of Iter	Time (sec.)	% (UB-LB)/LB	PSH	% (UB-LB)/LB
50	5	2	Low	1	344	301.085	302	31	3.38	0.00%	304	0.66%
				2	317	285.533	286	108	10.88	0.00%	288	0.70%
				3	361	301.233	302	34	3.57	0.00%	306	1.32%
				4	373	308.096	309	46	5.33	0.00%	310	0.32%
				5	320	281.088	282	29	2.70	0.00%	304	7.80%
				6	355	295.022	296	65	7.30	0.00%	313	5.74%
				7	321	285.121	286	51	4.93	0.00%	293	2.45%
				8	296	278.079	279	22	2.02	0.00%	282	1.08%
				9	355	301.988	302	113	12.16	0.00%	307	1.66%
				10	366	301.354	302	35	3.74	0.00%	302	0.00%
				11	312	281.01	282	41	3.78	0.00%	286	1.42%
				12	358	308.837	309	91	10.39	0.00%	310	0.32%
				13	300	278.31	279	84	7.85	0.00%	280	0.36%
				14	397	316.126	317	32	3.60	0.00%	320	0.95%
				15	359	301.458	302	30	3.24	0.00%	315	4.30%
				16	362	301.192	302	36	3.72	0.00%	311	2.98%
				17	330	289.151	290	28	2.80	0.00%	290	0.00%
				18	350	301.253	302	37	4.04	0.00%	302	0.00%
				19	331	289.108	290	26	2.27	0.00%	293	1.03%
				20	299	276.385	277	21	1.94	0.00%	284	2.53%
				21	290	277.579	279	149	14.05	0.36%	280	0.72%
				22	340	299.643	302	191	22.32	0.67%	310	3.33%
				23	321	287.659	290	177	18.14	0.69%	292	1.39%
				24	320	283.576	286	165	16.03	0.70%	289	1.76%
				25	335	283.414	286	170	18.05	0.70%	297	4.58%
				26	301	279.9	282	171	15.83	0.71%	282	0.71%
				27	349	305.095	309	163	19.04	0.98%	312	1.96%
				28	375	305.389	309	163	19.19	0.98%	340	11.11%
				29	341	298.997	302	167	17.75	1.00%	302	1.00%
				30	346	298.983	302	176	19.94	1.00%	303	1.34%
Average					-	-	-	88.40	9.33	0.26%	-	2.12%
50	5	2	High	1	295	278.041	279	52	6.68	0.00%	279	0.00%
				2	364	308.034	309	63	10.06	0.00%	309	0.00%
				3	342	295.862	296	133	19.80	0.00%	296	0.00%
				4	413	356.022	357	60	10.97	0.00%	357	0.00%
				5	401	334.98	335	113	20.40	0.00%	335	0.00%
				6	332	289.78	290	61	8.67	0.00%	290	0.00%
				7	322	285.04	286	57	8.33	0.00%	286	0.00%
				8	394	325.566	326	68	11.58	0.00%	327	0.31%
				9	286	276.767	277	183	22.94	0.00%	279	0.72%
				10	275	275	275	1	0.13	0.00%	275	0.00%
				11	378	316.323	317	37	6.05	0.00%	317	0.00%
				12	350	301.378	302	56	8.41	0.00%	302	0.00%
				13	355	301.9	302	79	11.98	0.00%	303	0.33%
				14	378	316.849	317	87	14.00	0.00%	317	0.00%
				15	385	316.505	317	59	9.46	0.00%	317	0.00%
				16	341	298.995	299	159	28.20	0.00%	302	1.00%
				17	351	308.281	310	168	28.87	0.32%	309	0.00%
				18	328	294.709	296	170	27.45	0.34%	296	0.34%
				19	299	280.366	282	172	24.32	0.36%	282	0.36%
				20	281	275.83	277	137	16.74	0.36%	277	0.36%
				21	327	287.604	290	172	26.43	0.69%	290	0.69%
				22	314	287.085	290	180	25.79	0.69%	290	0.69%
				23	302	279.873	282	152	24.82	0.71%	282	0.71%
				24	360	305.279	309	175	27.98	0.98%	309	0.98%
				25	360	305.251	309	171	29.17	0.98%	309	0.98%
				26	354	305.142	309	170	32.19	0.98%	309	0.98%
				27	360	305.142	309	166	28.49	0.98%	309	0.98%
				28	341	298.84	302	172	26.53	1.00%	302	1.00%
				29	351	298.998	302	175	29.70	1.00%	302	1.00%
				30	329	292.159	296	167	25.67	1.02%	296	1.02%
Average					-	-	-	120.5	19.06	0.35%	-	0.42%

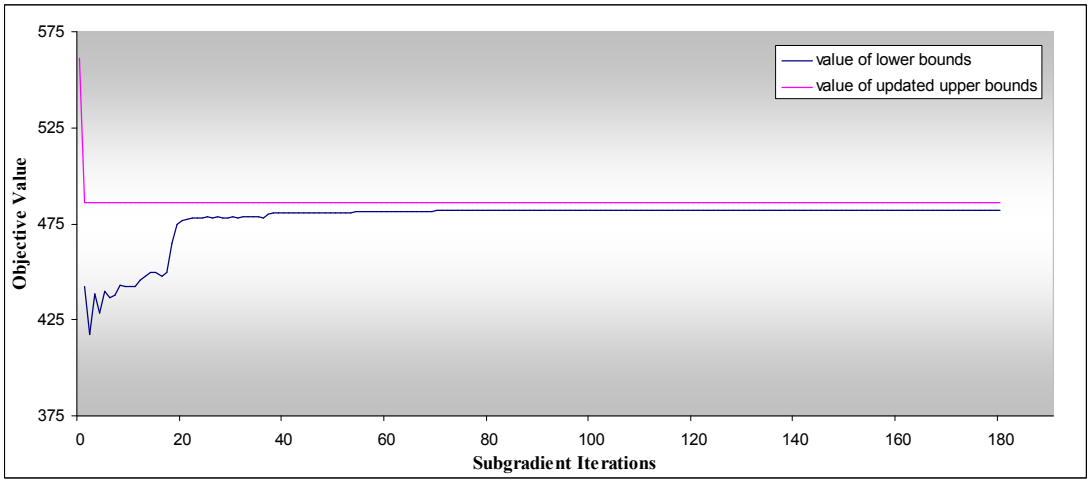
APPENDIX B. Computational results (LSA and PSH) - *Continues*

					Lagrangian-based Solution Approach (LSA)						Problem Specific Heuristic (PSH)	
					IH	LH						
n	m	b	F_p	Sample No		Lagr. LB	LH	# of Iter	Time (sec.)	% (UB-LB)/LB	PSH	% (UB-LB)/LB
100	5	2	Low	1	1324	1142.36	1144	177	96.52	0.09%	1155	1.05%
				2	1315	1130.24	1132	169	98.91	0.09%	1143	1.06%
				3	1343	1141.62	1145	181	103.75	0.26%	1148	0.53%
				4	1245	1108.37	1112	183	97.31	0.27%	1127	1.62%
				5	1210	1089.83	1093	173	104.98	0.28%	1092	0.18%
				6	1195	1080.01	1085	154	91.76	0.37%	1094	1.20%
				7	1339	1144	1149	177	86.77	0.44%	1160	1.40%
				8	1167	1078.49	1084	165	82.08	0.46%	1089	0.93%
				9	1342	1150.3	1157	158	93.25	0.52%	1160	0.78%
				10	1290	1114.66	1121	161	82.23	0.54%	1121	0.54%
				11	1201	1083.3	1090	149	80.12	0.55%	1085	0.09%
				12	1403	1164	1171	183	121.12	0.60%	1207	3.69%
				13	1348	1150.45	1158	167	100.31	0.61%	1158	0.61%
				14	1352	1150.33	1158	163	98.83	0.61%	1158	0.61%
				15	1276	1137.52	1145	158	95.56	0.62%	1151	1.14%
				16	1339	1137.96	1145	169	101.63	0.62%	1149	0.97%
				17	1300	1125.67	1133	163	88.79	0.62%	1160	3.02%
				18	1253	1100.27	1108	158	82.29	0.64%	1102	0.09%
				19	1178	1064.56	1072	160	99.46	0.66%	1089	2.25%
				20	1220	1102.95	1111	169	95.69	0.73%	1111	0.73%
				21	1430	1192.92	1202	167	108.22	0.75%	1208	1.26%
				22	1308	1120.46	1130	170	86.83	0.80%	1128	0.62%
				23	1206	1089.69	1099	164	74.01	0.83%	1098	0.73%
				24	1346	1156.62	1167	182	113.11	0.86%	1188	2.68%
				25	1226	1108.74	1119	168	91.64	0.90%	1115	0.54%
				26	1408	1177.86	1189	176	109.94	0.93%	1185	0.59%
				27	1311	1132	1144	164	97.03	1.06%	1137	0.44%
				28	1289	1132	1144	168	98.53	1.06%	1132	0.00%
				29	1291	1115.01	1128	166	96.56	1.08%	1128	1.08%
				30	1263	1100.86	1113	176	101.93	1.09%	1105	0.36%
Average					-	-	-	167.93	95.97	0.63%	-	1.03%
100	5	2	High	1	1297	1131.13	1132	65	51.78	0.00%	1132	0.00%
				2	1261	1104.77	1110	173	143.82	0.45%	1110	0.45%
				3	1310	1150.19	1157	171	158.79	0.52%	1157	0.52%
				4	1191	1086.03	1093	175	142.71	0.55%	1092	0.46%
				5	1421	1163.99	1171	165	160.29	0.60%	1171	0.60%
				6	1306	1132	1139	175	147.94	0.62%	1132	0.00%
				7	1106	1057.29	1065	166	125.28	0.66%	1061	0.28%
				8	1201	1083.02	1092	155	118.07	0.74%	1084	0.00%
				9	1162	1072.2	1081	179	132.27	0.75%	1077	0.37%
				10	1370	1157	1166	174	149.68	0.78%	1157	0.00%
				11	1173	1074.4	1085	171	120.03	0.93%	1077	0.19%
				12	1152	1067.46	1078	159	116.75	0.94%	1071	0.28%
				13	1128	1062.72	1074	156	113.14	1.03%	1065	0.19%
				14	1373	1192.41	1206	168	169.57	1.09%	1201	0.67%
				15	1241	1102.97	1116	162	135.79	1.18%	1110	0.63%
				16	1391	1184.99	1199	171	155.92	1.18%	1185	0.00%
				17	1211	1086.51	1100	175	133.35	1.20%	1092	0.46%
				18	1224	1086.1	1100	164	124.11	1.20%	1092	0.46%
				19	1196	1085.63	1099	165	110.19	1.20%	1092	0.55%
				20	1293	1156	1170	173	147.94	1.21%	1157	0.09%
				21	1146	1067.82	1081	160	120.00	1.22%	1071	0.28%
				22	1315	1137.59	1153	161	144.27	1.32%	1144	0.53%
				23	1249	1109.28	1125	175	136.18	1.35%	1110	0.00%
				24	1227	1103.24	1119	161	134.28	1.36%	1110	0.54%
				25	1183	1080.77	1096	151	119.13	1.39%	1084	0.28%
				26	1215	1089.84	1106	161	116.84	1.47%	1092	0.18%
				27	1204	1089.94	1106	156	119.06	1.47%	1092	0.18%
				28	1273	1119.44	1137	157	136.13	1.52%	1121	0.09%
				29	1257	1113.49	1131	177	153.06	1.53%	1133	1.71%
				30	1132	1130.39	1149	164	133.47	1.59%	1132	0.09%
Average					-	-	-	162.83	132.33	1.03%	-	0.34%

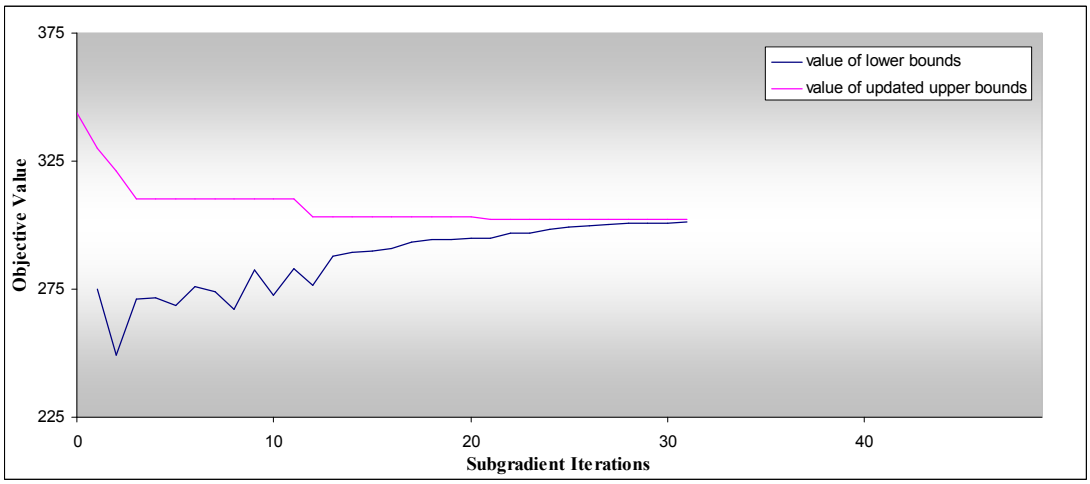
APPENDIX B. Computational results (LSA and PSH) - *Continues*

					Lagrangian-based Solution Approach (LSA)						Problem Specific Heuristic (PSH)	
					IH	LH						
n	m	b	F_p	Sample No		Lagr. LB	LH	# of Iter	Time (sec.)	% (UB-LB)/LB	PSH	% (UB-LB)/LB
100	8	3	Low	1	909	747.095	748	29	16.92	0.00%	750	0.27%
				2	877	747.037	748	35	21.81	0.00%	753	0.67%
				3	850	723.944	724	86	46.38	0.00%	727	0.41%
				4	814	723.239	724	82	43.73	0.00%	729	0.69%
				5	918	766.791	769	172	101.53	0.26%	767	0.00%
				6	834	731.286	734	158	87.65	0.27%	740	1.09%
				7	822	720.113	724	153	87.41	0.42%	727	0.83%
				8	832	716.963	720	165	86.00	0.42%	720	0.42%
				9	813	713.941	717	164	105.23	0.42%	717	0.42%
				10	791	707.554	711	163	92.38	0.42%	712	0.56%
				11	774	696.538	700	158	83.63	0.43%	704	1.00%
				12	882	752.476	757	173	106.69	0.53%	759	0.80%
				13	862	743.931	748	169	104.76	0.54%	749	0.67%
				14	843	743.201	748	154	104.71	0.54%	749	0.67%
				15	862	743.988	748	168	101.82	0.54%	748	0.54%
				16	887	743.685	748	159	93.29	0.54%	761	2.28%
				17	844	727.597	732	164	96.85	0.55%	732	0.55%
				18	833	723.507	728	163	88.33	0.55%	726	0.28%
				19	793	715.18	720	162	89.25	0.56%	719	0.42%
				20	794	713.451	718	160	87.11	0.56%	719	0.70%
				21	942	782.498	788	171	111.40	0.64%	789	0.77%
				22	897	771.986	777	174	105.56	0.65%	777	0.65%
				23	914	771.836	777	174	108.64	0.65%	779	0.91%
				24	834	727.627	733	158	92.67	0.69%	734	0.82%
				25	840	727.364	734	166	99.00	0.82%	735	0.96%
				26	803	715.707	722	160	87.86	0.84%	717	0.14%
				27	780	697.203	704	174	107.34	0.86%	723	3.58%
				28	966	776.719	784	169	94.28	0.90%	779	0.26%
				29	909	756.997	765	180	107.00	1.06%	770	1.72%
				30	851	731.35	740	159	92.85	1.09%	740	1.09%
Average					-	-	-	150.73	88.40	0.52%	-	0.81%
100	8	3	High	1	858	723.997	724	146	110.01	0.00%	724	0.00%
				2	897	747.171	748	45	37.30	0.00%	748	0.00%
				3	918	776.214	777	60	55.42	0.00%	777	0.00%
				4	934	776.12	777	63	53.76	0.00%	777	0.00%
				5	942	776.004	777	61	57.38	0.00%	777	0.00%
				6	891	747.099	748	59	44.56	0.00%	748	0.00%
				7	878	747.629	748	75	66.53	0.00%	748	0.00%
				8	855	739.995	742	175	155.29	0.27%	740	0.00%
				9	873	739.476	742	183	145.46	0.27%	740	0.00%
				10	965	798.609	802	176	147.17	0.38%	799	0.00%
				11	843	720.351	724	159	130.43	0.42%	724	0.42%
				12	786	707.515	711	163	115.56	0.42%	711	0.42%
				13	772	701.102	705	172	127.92	0.43%	705	0.43%
				14	750	691.528	695	151	149.87	0.43%	695	0.43%
				15	891	752.211	757	165	144.14	0.53%	757	0.53%
				16	923	752.241	757	171	159.35	0.53%	757	0.53%
				17	844	735.237	740	162	143.40	0.54%	740	0.54%
				18	855	735.999	740	165	145.90	0.54%	740	0.54%
				19	855	731.755	736	168	143.60	0.55%	732	0.00%
				20	843	723.79	728	166	131.81	0.55%	732	1.10%
				21	832	723.358	728	151	125.88	0.55%	724	0.00%
				22	944	771.636	777	174	164.49	0.65%	777	0.65%
				23	840	743.708	749	167	145.47	0.67%	748	0.54%
				24	890	756.998	763	168	154.61	0.79%	757	0.00%
				25	833	731.08	738	169	147.53	0.82%	732	0.00%
				26	802	712.865	719	162	133.54	0.84%	717	0.56%
				27	867	735.819	743	170	150.33	0.95%	740	0.54%
				28	965	809.968	818	170	152.90	0.99%	810	0.00%
				29	775	700.992	708	155	103.68	1.00%	705	0.57%
				30	767	696.752	704	149	107.26	1.00%	700	0.43%
Average					-	-	-	144.00	121.68	0.47%	-	0.27%

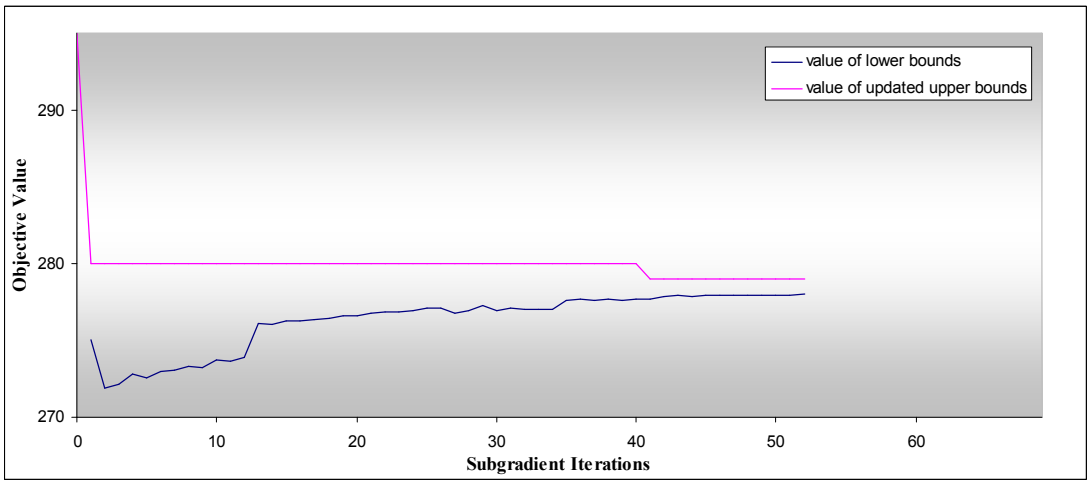
APPENDIX C. Sample convergence graphics related to each sub-group of test problems



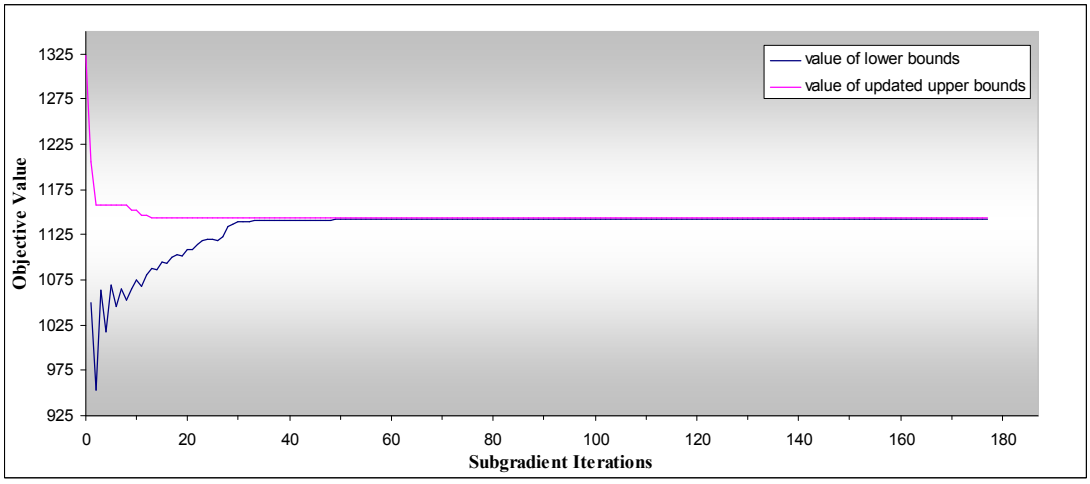
APPENDIX C1. Convergence representation of 50-3-1-High-Sample 1



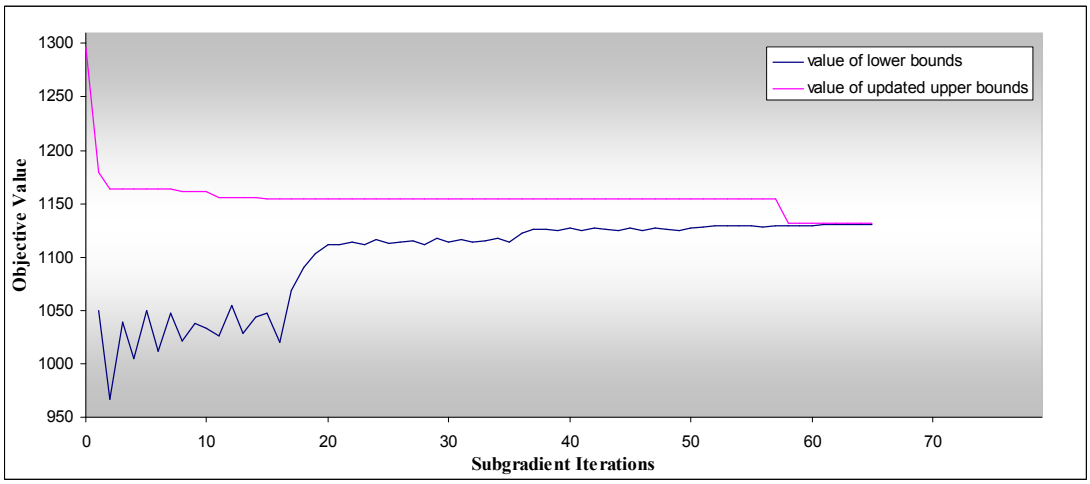
APPENDIX C2. Convergence Representation of 50-5-2-Low-Sample 1



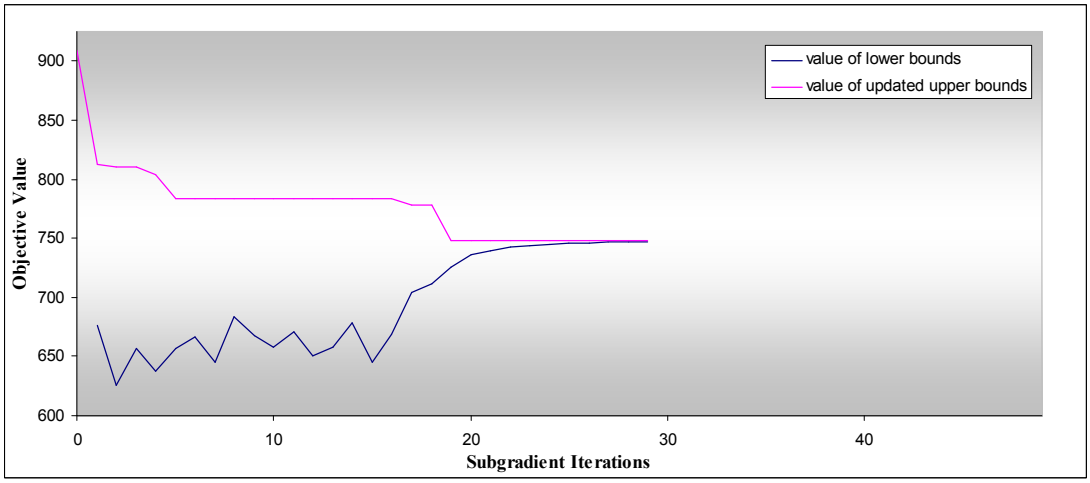
APPENDIX C3. Convergence representation of 50-5-2-High-Sample 1



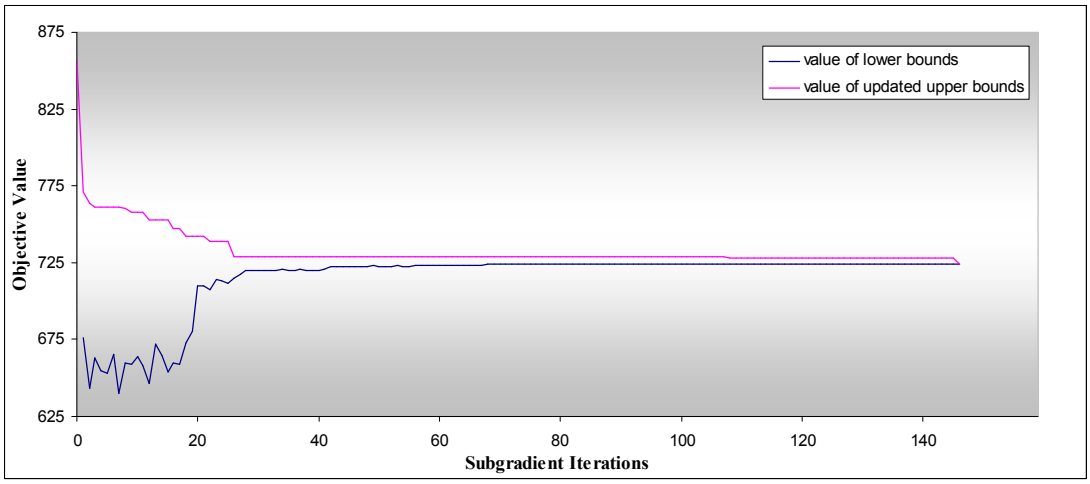
APPENDIX C4. Convergence representation of 100-5-2-Low-Sample 1



APPENDIX C5. Convergence representation of 100-5-2-High-Sample 1



APPENDIX C6. Convergence representation of 100-8-3-Low-Sample 1



APPENDIX C7. Convergence representation of 100-8-3-High-Sample 1

APPENDIX D1 Computational results of 30-job test instances

F_P	F_B	*Samp. No*	*# of Periods (T)*	IP			CP			IP/CP COMBINED		
				Obj. Value	**CPU Time (*s*)**	**Gap %**	**Obj. Value**	**CPU Time (*s*)**	**Gap %**	**Obj. Value**	**CPU Time (*s*)**	**Gap %**
Low	Low	1	74	71	349.25	0.00	71	83.68	0.00	71	22.28	0.00
		2	81	72	324.22	0.00	74	1000.00*	2.78	72	109.51	0.00
		3	64	56	1000.00*	1.82	55	100.18	0.00	55	52.87	0.00
		4	60	59	1000.00*	1.72	58	0.09	0.00	58	10.87	0.00
		5	72	66	1000.00*	3.13	66	1000.00*	3.13	**64**	**91.16**	**0.00**
		6	77	74	75.66	0.00	75	1000.00*	1.35	74	35.21	0.00
		7	74	72	905.02	0.00	72	0.07	0.00	72	15.53	0.00
		8	78	74	133.27	0.00	75	1000.00*	1.35	74	31.10	0.00
		9	66	62	44.66	0.00	64	1000.00*	3.23	62	79.85	0.00
		10	73	67	159.01	0.00	67	5.40	0.00	67	31.62	0.00
		11	81	74	103.49	0.00	75	1000.00*	1.35	74	39.65	0.00
		12	89	85	63.91	0.00	87	1000.00*	2.35	85	104.35	0.00
Low	High	1	74	67	31.00	0.00	67	66.35	0.00	67	46.80	0.00
		2	65	60	137.27	0.00	61	1000.00*	1.67	60	19.48	0.00
		3	85	78	38.01	0.00	78	148.60	0.00	78	43.24	0.00
		4	67	64	1000.00*	1.59	64	1000.00*	1.59	**63**	**28.79**	**0.00**
		5	71	70	6.05	0.00	70	0.03	0.00	70	16.48	0.00
		6	83	76	35.14	0.00	77	1000.00*	1.32	76	53.57	0.00
		7	79	77	27.69	0.00	77	7.13	0.00	77	18.12	0.00
		8	78	75	1000.00*	1.35	74	1.35	0.00	74	38.73	0.00
		9	72	64	417.45	0.00	64	41.85	0.00	64	33.17	0.00
		10	66	59	108.00	0.00	60	1000.00*	1.69	59	42.53	0.00
		11	71	62	124.13	0.00	63	1000.00*	1.61	62	54.84	0.00
		12	84	77	138.81	0.00	80	1000.00*	3.90	77	65.76	0.00
High	Low	1	64	62	7.81	0.00	63	1000.00*	1.61	62	8.43	0.00
		2	62	60	12.35	0.00	60	0.03	0.00	60	9.20	0.00
		3	69	66	26.04	0.00	66	2.45	0.00	66	12.77	0.00
		4	60	58	342.28	0.00	58	0.02	0.00	58	8.79	0.00
		5	69	68	11.44	0.00	68	0.29	0.00	68	8.62	0.00
		6	79	78	1000.00*	1.30	78	1000.00*	1.30	77	39.22	0.00
		7	66	63	26.28	0.00	65	1000.00*	3.17	63	13.40	0.00
		8	66	65	113.31	0.00	-	1000.00*	-	65	20.11	0.00
		9	66	64	127.18	0.00	64	238.85	0.00	64	10.80	0.00
		10	69	67	1000.00*	1.52	67	1000.00*	1.52	**66**	**18.41**	**0.00**
		11	77	76	170.35	0.00	76	7.00	0.00	76	12.89	0.00
		12	69	64	68.27	0.00	65	0.02	1.56	64	12.51	0.00
High	High	1	78	70	1000.00*	1.45	69	0.88	0.00	69	23.20	0.00
		2	75	70	547.91	0.00	70	44.19	0.00	70	18.81	0.00
		3	74	67	91.83	0.00	67	201.18	0.00	67	25.31	0.00
		4	68	-	1000.00*	-	68	1000.00*	1.49	**67**	**15.35**	**0.00**
		5	69	67	1000.00*	1.52	66	21.09	0.00	66	11.50	0.00
		6	65	64	304.93	0.00	64	2.27	0.00	64	9.94	0.00
		7	60	57	27.60	0.00	57	0.02	0.00	57	8.63	0.00
		8	75	74	13.05	0.00	74	1.63	0.00	74	9.79	0.00
		9	86	77	39.71	0.00	77	72.08	0.00	77	31.83	0.00
		10	67	65	48.80	0.00	65	32.21	0.00	65	11.32	0.00
		11	72	64	52.19	0.00	64	0.04	0.00	64	23.72	0.00
		12	80	77	20.71	0.00	78	1000.00*	1.30	77	21.97	0.00

* Run is aborted due to run limit(1000 seconds)

Gray shaded cells: The corresponding model dominates other two models in terms of CPU time and/or objective function value.

Values in Bold: The corresponding model provides better makespan values than other two models.

APPENDIX D2 Computational results of 50-job test instances

F_P	F_B	*Samp. No*	*# of Periods (T)*	IP			CP			IP/CP COMBINED		
				Obj. Value	CPU Time (*s*)	Gap %	Obj. Value	CPU Time (*s*)	Gap %	Obj. Value	CPU Time (*s*)	Gap %
Low	Low	1	81	73	1000.00*	1.39	73	1000.00*	1.39	**72**	**518.43**	**0.00**
		2	81	75	834.52	0.00	75	0.10	0.00	75	404.75	0.00
		3	68	67	1000.00*	1.52	67	1000.00*	1.52	**66**	**150.38**	**0.00**
		4	75	70	1000.00*	0.00	70	1000.00*	0.00	70	1000.00*	0.00
		5	77	**72**	**715.95**	**0.00**	75	1000.00*	4.17	73	1000.00*	1.39
		6	84	82	105.71	0.00	83	1000.00*	1.22	82	235.08	0.00
		7	79	75	287.21	0.00	76	1000.00*	1.33	75	307.60	0.00
		8	80	79	247.00	0.00	79	0.42	0.00	79	156.69	0.00
		9	81	73	1000.00*	2.82	72	1000.00*	1.41	**71**	**566.16**	**0.00**
		10	74	72	276.08	0.00	73	1000.00*	1.39	72	182.60	0.00
		11	80	71	1000.00*	1.43	70	187.06	0.00	70	599.40	0.00
		12	91	88	1000.00*	1.15	88	1000.00*	1.15	**87**	**372.09**	**0.00**
Low	High	1	79	70	1000.00*	1.45	69	0.11	0.00	69	519.92	0.00
		2	78	74	276.07	0.00	75	1000.00*	1.35	74	310.02	0.00
		3	83	74	381.26	0.00	78	1000.00*	5.41	74	669.64	0.00
		4	79	76	1000.00*	1.33	75	379.57	0.00	75	357.62	0.00
		5	91	88	1000.00*	1.15	87	1.82	0.00	87	362.26	0.00
		6	90	84	1000.00*	2.44	84	1000.00*	2.44	**82**	**759.98**	**0.00**
		7	77	72	142.70	0.00	73	1000.00*	1.39	72	332.91	0.00
		8	79	79	1000.00*	1.28	79	1000.00*	1.28	**78**	**152.30**	**0.00**
		9	83	-	1000.00*	-	82	1000.00*	1.23	**81**	**203.74**	**0.00**
		10	86	83	1000.00*	1.22	82	682.57	0.00	82	372.97	0.00
		11	78	73	1000.00*	1.39	73	1000.00*	1.39	**72**	**443.19**	**0.00**
		12	75	72	1000.00*	1.41	72	1000.00*	1.41	**71**	**213.94**	**0.00**
High	Low	1	74	72	222.31	0.00	74	1000.00*	2.78	72	112.19	0.00
		2	68	67	1000.00*	1.52	67	1000.00*	1.52	**66**	**86.24**	**0.00**
		3	76	72	1000.00*	1.41	71	36.11	0.00	71	188.41	0.00
		4	77	76	1000.00*	1.33	76	1000.00*	1.33	**75**	**122.13**	**0.00**
		5	74	-	1000.00*	-	72	1000.00*	1.41	**71**	**113.94**	**0.00**
		6	73	70	1000.00*	1.45	70	1000.00*	1.45	**69**	**154.56**	**0.00**
		7	70	-	1000.00*	-	69	0.10	0.00	69	84.29	0.00
		8	80	72	1000.00*	2.86	72	1000.00*	2.86	**70**	**449.28**	**0.00**
		9	68	67	1000.00*	1.52	66	0.13	0.00	66	90.62	0.00
		10	71	65	1000.00*	3.17	64	1000.00*	1.59	**63**	**171.61**	**0.00**
		11	82	81	556.69	0.00	81	0.08	0.00	81	117.70	0.00
		12	73	71	686.05	0.00	72	1000.00*	1.41	**71**	**119.95**	**0.00**
High	High	1	77	68	1000.00*	1.49	69	1000.00*	2.99	**67**	**298.58**	**0.00**
		2	73	70	844.93	0.00	73	1000.00*	4.29	70	137.61	0.00
		3	89	81	1000.00*	1.25	82	1000.00*	2.50	**80**	**462.06**	**0.00**
		4	71	69	1000.00*	1.47	68	62.54	0.00	68	109.82	0.00
		5	73	71	76.53	0.00	71	3.37	0.00	71	94.44	0.00
		6	81	78	992.69	0.00	79	1000.00*	1.28	78	160.35	0.00
		7	80	72	171.05	0.00	72	0.13	0.00	72	263.19	0.00
		8	82	80	1000.00*	1.27	79	0.07	0.00	79	124.03	0.00
		9	80	75	907.06	0.00	76	1000.00*	1.33	75	197.74	0.00
		10	84	82	500.13	0.00	83	1000.00*	1.22	82	152.25	0.00
		11	82	79	796.90	0.00	79	1.30	0.00	79	149.27	0.00
		12	92	86	816.56	0.00	87	1000.00*	1.16	86	279.95	0.00

* Run is aborted due to run limit(1000 seconds)

Gray shaded cells: The corresponding model dominates other two models in terms of CPU time and/or objective function value.

Values in Bold: The corresponding model provides better makespan values than other two models.