

CUT GENERATION BASED ALGORITHMS FOR  
UNRELATED PARALLEL MACHINE SCHEDULING PROBLEMS

by  
HALİL ŞEN

Submitted to the Graduate School of Engineering and Natural Sciences  
in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy

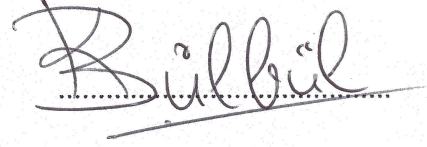
Sabancı University, İstanbul

August 2015

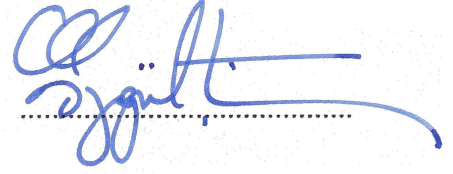
CUT GENERATION BASED ALGORITHMS FOR  
UNRELATED PARALLEL MACHINE SCHEDULING PROBLEMS

APPROVED BY

Assoc. Prof. Dr. Kerem Bülbül  
(Dissertation Supervisor)



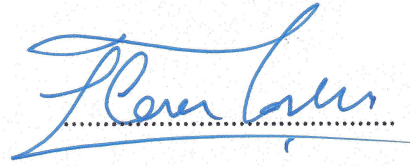
Prof. Dr. Özgür Erçetin



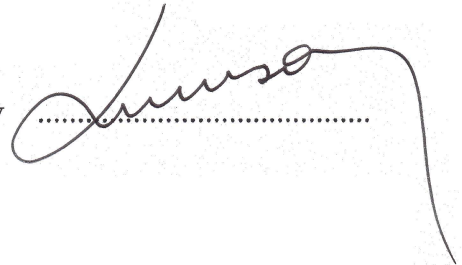
Prof. Dr. Ceyda Oğuz



Assoc. Prof. Dr. Z. Caner Taşkın



Prof. emer. Dr. Gündüz A. Ulusoy



DATE OF APPROVAL: 21/07/2015

©Halil Şen, 2015  
All Rights Reserved

## ABSTRACT

### CUT GENERATION BASED ALGORITHMS FOR UNRELATED PARALLEL MACHINE SCHEDULING PROBLEMS

HALİL ŞEN

Ph.D. Dissertation, August 2015

Dissertation Supervisor: Assoc. Prof. Dr. Kerem Bülbül

Keywords: unrelated parallel machine scheduling, Benders decomposition,  
logic-based Benders decomposition, exact method, heuristics

Research on scheduling in the unrelated parallel machine environment is at best scarce. Moreover, almost all existing work in this area is focused on the minimization of completion time related performance measures and the solution approaches available in the literature suffer from scalability issues. In this dissertation, we leverage on the success of the mathematical programming based decomposition approaches and devise scalable, efficient, and effective cut generation based algorithms for four  $\mathcal{NP}$ -hard unrelated parallel machine scheduling problems.

In the first part, we develop a new preemptive relaxation for the total weighted tardiness and total weighted earliness/tardiness problems and devise a Benders decomposition algorithm for solving this preemptive relaxation formulated as a mixed integer linear program. We demonstrate the effectiveness of our approach with instances up to 5 machines and 200 jobs.

The second part deals with the problem of minimizing the total weighted completion time and proves that the preemptive relaxation developed in part one is an exact formulation for this problem. By exploiting the structural properties of the performance measure, we attain an exact Benders decomposition algorithm which solves instances with up to 1000 jobs and 8 machines to optimality within a few seconds.

In the last part, we tackle the unrestricted common due date just-in-time scheduling problem and develop a logic-based Benders decomposition algorithm. Aside from offering the best solution approach for this problem, we demonstrate that it is possible to devise scalable logic-based algorithms for scheduling problems with irregular minsum objectives.

## ÖZET

### ALAKASIZ PARALEL MAKİNE ÇİZELGELEME PROBLEMLERİNE KESİ TÜRETME TABANLI ALGORİTMALAR

HALİL ŞEN

Doktora Tezi, Ağustos 2015

Tez Danışmanı: Doç. Dr. Kerem Bülbül

Anahtar Kelimeler: alakasız paralel makine çizelgeleme, Benders ayrıştırma, mantık tabanlı Benders ayrıştırma, pekin yöntem, sezgisel yöntem

Alakasız paralel makine ortamındaki çizelgeleme problemleri üzerindeki araştırmalar en iyimser bakış açısıyla sınırlı durumdadırlar. Dahası, bu alanda var olan çalışmaların neredeyse tümü iş tamamlanma zamanıyla alakalı performans ölçütlerinin enküçüklenmesine yoğunlaşmış durumdadır ve literatürdeki mevcut çözüm yaklaşımları ölçeklenebilirlik sorunlarından muzdaripdirler. Bu tezde, matematiksel programlama tabanlı ayrıştırma yaklaşımlarının başarısından güç alınarak, dört adet  $NP$ -zor alakasız paralel makine çizelgeleme problemine ölçeklenebilir, etkili ve yüksek verimli kesi türetme tabanlı algoritmalar tasarlanmıştır.

İlk kısımda, toplam ağırlıklandırılmış gecikme ve toplam ağırlıklandırılmış erkenlik/gecikme problemleri için yeni bir geçişli gevşetme geliştirilmiş ve bir karışık tamsayılı doğrusal program olarak formüle edilmiş bu geçişli gevşetmeyi çözmek için bir Bender ayrıştırma algoritması tasarlanmıştır. Yaklaşımımızın etkinliğini göstermek üzere 5 makine ve 200 iş büyüklüğüne kadar örnekler çözülmüştür.

İkinci kısım toplam ağırlıklandırılmış tamamlanma zamanı problemini ele almakta ve ilk kısımda geliştirilen geçişli gevşetmenin bu problem için pekin bir gösterim olduğunu ispatlamaktadır. Dahası, bu performans ölçütünün yapısal özelliklerinden faydalanılarak, 8 makine ve 1000 iş büyüklüğüne kadar örneklerin eniyi çözümlerine saniyeler içerisinde ulaşan pekin bir Benders ayrıştırma algoritması elde edilmiştir.

Sonuncu kısımda ise kısıtlandırılmamış ortak termin zamanlı tam zamanında çizelgeleme problemi ele alınmakta ve mantık tabanlı Benders ayrıştırma algoritması geliştirilmektedir. Bu problem için en başarılı çözüm yaklaşımını sunmanın yanı sıra, bu kısım, düzenli olmayan enküçük-toplam performans ölçütlü çizelgeleme problemleri için ölçeklenebilir bir mantık tabanlı algoritma tasarlanmasının mümkün olduğunu göstermektedir.

*to Babik, my love and muse*  
*and*  
*to my friends and parents*

## Acknowledgments

First and foremost I would like to express my gratitude and appreciation to my advisor, Kerem Bülbül, who continued to believe in me despite me not giving him a reason to. I have been lucky enough to work with him, and without his substantial help, support and encouragement, this dissertation would not have been possible.

I would like to express my deepest love and gratitude to Babik, without whom I wouldn't be the person I am today. Her love, support, and encouragement nurtured my spirit and soul. I will be eternally in her debt. I would also like to thank İnci and Öncel Koca who opened their home to me and treated me as one of their own.

Many thanks to all of my friends for being there whenever I needed, and for tolerating me at my bad moments. I cannot possibly state all, but I would like to mention some of them for sharing important moments of my life. Belit and Ozan Dağdeviren, Sezen and Anıl Can, Nurşen and Ömer M. Özkırmı, Neva Özcü, Selin Erçil, Arda Kurtoğlu, U. Mahir Yıldırım, Gülnur Kocapınar, Belma Yelbay, Koray Kuvvet, and Funda Aktan, I am very happy to have you in my life. I would not be able to keep my sanity if it weren't for you.

I also would like to thank the people of VanDerSal – especially, Barış Özgür Çıtır, Eren Kozluca, Mert Soykan, and İlhan Şahiner – for the occasionally wasteful, but nevertheless wonderful times we have spent all together.

Many thanks to Sinem Aydın, Banu Akıncı, Barış Tümer, and Osman Rahmi Fıçıcı for all of their help with the bureaucratic and technical university matters.

Finally, I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for financially supporting me throughout my graduate life within the frameworks of the 2210- and 2211-National Scholarship Programme for Graduate Students.

# TABLE OF CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Outline . . . . .	3
<b>2</b>	<b>A STRONG PREEMPTIVE RELAXATION FOR TOTAL WEIGHTED TARDINESS AND EARLINESS/TARDINESS</b>	<b>5</b>
2.1	Introduction . . . . .	6
2.2	Review of Related Literature . . . . .	8
2.3	Problem Statement and Preemptive Relaxation . . . . .	17
2.4	Benders Decomposition . . . . .	23
2.4.1	Validity and Strengthening of the Benders Cuts . . . . .	26
2.5	Computational Results . . . . .	32
2.5.1	Results for <i>Rm-TWT</i> . . . . .	35
2.5.2	Results for <i>Rm-TWET</i> . . . . .	42
<b>3</b>	<b>AN EXACT EXTENDED FORMULATION FOR TOTAL WEIGHTED COMPLETION TIME</b>	<b>48</b>
3.1	Introduction . . . . .	49
3.2	Review of Related Literature . . . . .	51
3.3	Formulation and Solution Approach . . . . .	54
3.3.1	Benders Decomposition . . . . .	62
3.4	Computational Results . . . . .	72
<b>4</b>	<b>LOGIC-BASED BENDERS DECOMPOSITION FOR COMMON DUE DATE TOTAL WEIGHTED EARLINESS/TARDINESS</b>	<b>80</b>
4.1	Introduction . . . . .	81
4.2	Review of Related Literature . . . . .	82
4.2.1	Parallel Machine Scheduling . . . . .	83
4.2.2	LBBD in Scheduling . . . . .	85
4.3	Solution Approach . . . . .	87
4.3.1	Overview of LBBD . . . . .	88
4.3.2	LBBD for <i>Rm-UCDD</i> . . . . .	90
4.3.3	Strengthened Bounding Functions . . . . .	92
4.4	Computational Results . . . . .	101
<b>5</b>	<b>CONCLUSION AND FUTURE RESEARCH</b>	<b>109</b>
	<b>Bibliography</b>	<b>111</b>



# LIST OF TABLES

2.1	Summary of the important points in Section 2.2. . . . .	15
2.2	Instance generation parameters. . . . .	33
2.3	Results for $Rm-TWT$ . . . . .	36
2.4	Results for $Rm-TWET$ . . . . .	44
3.1	Average optimality gap and solution time results for $Rm-TWCT$ . . .	75
4.1	Average optimality gap and solution time results for $Rm-UCDD$ . .	103

# LIST OF FIGURES

2.1	The empirical distributions of the optimality gaps of the upper bounds by <b>(TR – A)-BDS</b> for <i>Rm-TWT</i> . . . . .	39
2.2	The empirical distributions of the solution times of <b>(TR – A)-BDS</b> and <b>(TR – A)-CPX</b> for <i>Rm-TWT</i> . . . . .	41
2.3	The empirical distributions of the optimality gaps of the upper bounds by <b>(TR – A)-BDS</b> for <i>Rm-TWET</i> . . . . .	45
2.4	The empirical distributions of the solution times of <b>(TR – A)-BDS</b> and <b>(TR – A)-CPX</b> for <i>Rm-TWET</i> . . . . .	46
3.1	The empirical distributions of the solution times and the optimality gaps of <b>(TR – A)-BDS</b> (—) and <b>(CQ)-CPLEX</b> (– –) for <i>Rm-TWCT</i> instances with 2, 4, and 6 machines. . . . .	77
3.2	The empirical distributions of the solution times and the optimality gaps of <b>(TR – A)-BDS</b> (—) and <b>(CQ)-CPLEX</b> (– –) for <i>Rm-TWCT</i> instances with 8, 16, and 30 machines. . . . .	78
4.1	The empirical distributions of the solution times and the optimality gaps of <b>BDS</b> (—) and <b>CPX</b> (– –) for <i>Rm-UCDD</i> instances with 50, 60, and 80 jobs. . . . .	106
4.2	The empirical distributions of the solution times and the optimality gaps of <b>BDS</b> (—) and <b>CPX</b> (– –) for <i>Rm-UCDD</i> instances with 100, 400, and 1000 jobs. . . . .	107

# LIST OF ABBREVIATIONS

Notation	Description	Page List
B&B	Branch-and-bound	2, 6, 8, 9, 15, 51, 52, 109
CG	Column generation	10, 52, 86
CP	Constraint programming	84, 85, 86
CQIP	Convex quadratic integer programming	15, 52, 54, 72, 81, 84, 88, 90, 100
LBBD	Logic-based Benders decomposition	2, 4, 80, 81, 82, 84, 86, 88, 89, 90, 99, 105, 110
LP	Linear programming	6, 10, 12, 19, 22, 23, 38, 40, 41, 43, 50, 52, 54, 57, 61, 62, 63, 83, 84, 85, 89
LR	Lagrangian relaxation	1, 10, 11, 12, 15
MIP	Mixed integer linear programming	2, 3, 5, 6, 22, 25, 48, 49, 50, 52, 54, 57, 84, 85, 86, 109
$Rm\text{-UCDD}$	$Rm/d_j = d^l / \sum_j \epsilon_j E_j + \pi_j T_j$	81, 82, 83, 84, 87, 88, 89, 90, 100, 104, 105, 108, 110
$Rm\text{-TWCT}$	$Rm // \sum_j w_j C_j$	48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 61, 64, 69, 72, 74, 76, 109
$Rm\text{-TWET}$	$Rm // \sum_j \epsilon_j E_j + \pi_j T_j$	5, 6, 12, 17, 18, 19, 21, 22, 24, 25, 26, 28, 30, 32, 33, 34, 35, 40, 41, 42, 43, 45, 46, 49, 50, 57, 58, 109, 110
$Rm\text{-TWT}$	$Rm // \sum_j \pi_j T_j$	5, 6, 12, 17, 18, 19, 21, 22, 23, 24, 25, 26, 30, 31, 32, 33, 34, 35, 37, 40, 41, 42, 43, 45, 46, 49, 50, 57, 58, 109, 110

<b>Notation</b>	<b>Description</b>	<b>Page List</b>
TWCT	Total weighted completion time	<a href="#">3</a> , <a href="#">4</a> , <a href="#">48</a> , <a href="#">50</a> , <a href="#">51</a> , <a href="#">57</a> , <a href="#">61</a> , <a href="#">62</a> , <a href="#">68</a> , <a href="#">84</a>
TWET	Total weighted earliness/tardiness	<a href="#">3</a> , <a href="#">5</a> , <a href="#">6</a> , <a href="#">47</a> , <a href="#">49</a> , <a href="#">50</a> , <a href="#">53</a>
TWT	Total weighted tardiness	<a href="#">3</a> , <a href="#">5</a> , <a href="#">6</a> , <a href="#">9</a> , <a href="#">49</a> , <a href="#">50</a> , <a href="#">53</a>
UCDD	Unrestrictive common due date total weighted earliness/tardiness	<a href="#">3</a> , <a href="#">4</a> , <a href="#">80</a> , <a href="#">90</a>
WLPT	Weighted longest processing time	<a href="#">90</a> , <a href="#">100</a>
WSPT	Weighted shortest processing time	<a href="#">49</a> , <a href="#">55</a> , <a href="#">59</a> , <a href="#">61</a> , <a href="#">65</a> , <a href="#">68</a> , <a href="#">69</a> , <a href="#">70</a> , <a href="#">71</a> , <a href="#">90</a> , <a href="#">100</a>

## CHAPTER 1

# INTRODUCTION

The prevalence of actual manufacturing environments where a set of tasks has to be executed on a set of alternate resources attests to the practical relevance of the parallel machine scheduling environment. For instance, many production steps in semiconductor manufacturing feature unrelated parallel machines because existing machines are augmented over time with machines of newer technology for ramping up production (Shim and Kim, 2007a). Another setting observed in the inspection operations in semiconductor manufacturing creates the context for a recent work by Detienne et al. (2011) on unrelated parallel machines with step-wise individual job cost functions. Several other industries, such as the beverage, printing, and pharmaceutical industries, require processing steps performed by a set of parallel machines (Biskup et al., 2008). Therefore, a thorough understanding of the trade-offs that govern the parallel machine environment is fundamental for the successful operation in many different manufacturing settings.

The scheduling literature is often criticized for its emphasis on the single-machine environment which is arguably not encountered frequently in today's complex shop floors. However, virtually every scheduling algorithm conceived for multi-stage production systems does either generalize or depend upon the fundamental principles derived from the basic single-machine scheduling problems. A similar argument is valid for the parallel machine environment as well (Pinedo, 2008, p.111). Decomposition algorithms devised for multi-stage systems, such as Lagrangian relaxation (LR), Dantzig-Wolfe reformulation, Benders decomposition, and the shifting bottleneck heuristic, give rise to either single- or parallel machine scheduling subproblems that have to be solved many times in an iter-

ative framework. The ultimate performance of such decomposition approaches depends critically on our ability to solve these machine scheduling subproblems with a high solution quality in short computational times. Moreover, the study of parallel machines is the immediate logical extension of single-machine scheduling from a theoretical perspective and for a given partition of the set of jobs over the set of machines, a parallel machine scheduling problem is just a collection of independent single-machine scheduling problems. Therefore, parallel machine scheduling problems are generally regarded as set partitioning problems where the complexity of calculating the cost of a partition depends on the difficulty of the underlying single-machine scheduling problem. Furthermore, our specific interest in unrelated parallel machines is also prompted by a simple observation – i.e., capacity expansions over time naturally result in production steps performed on a set of unrelated parallel machines as equipment technology evolves. [Shim and Kim \(2007a\)](#), for instance, discuss this issue in the context of semiconductor manufacturing. Thus, there is a clear need for good algorithms tailored to the unrelated parallel machine environment. However, the literature reviews presented in Sections [2.2](#), [3.2](#), and [4.2](#) reveal that the research on unrelated parallel machine scheduling is at best scarce in the fullest sense of the word.

Multi-machine scheduling problems may be modeled as mixed integer linear programming (MIP) problems in which all necessary decisions – e.g., assignment, sequencing, scheduling – are handled simultaneously by a monolithic formulation. However, exploiting the aforementioned set partitioning nature of the problem and separating some of these decisions from each other and tackling them synchronously may go a long way in terms of computational efficiency. It turns out that similar exact solution procedures – that rely on mathematical programming based decomposition techniques – proposed for identical parallel machine scheduling problems and their performances are far more promising compared to those of monolithic formulations and custom branch-and-bound (B&B) procedures. Furthermore, this is true for both completion time and the due date related performance measures.

Another line of successful solution methods for multi-machine scheduling problems is due to the logic-based Benders decomposition (LBBD) framework developed in recent years. The basic principle of this framework is to find a

valid *bounding function* which represents a lower bound on the optimal objective function value of the problem. Then, this bounding function is used to create cuts in the Benders decomposition algorithm. The efficacy of this type of solution procedure is demonstrated in the literature on several scheduling problems with basic objective functions – such as finding a feasible solution, minimizing job to machine assignment costs, etc. – and on those with regular minmax performance measures – e.g., minimizing makespan, maximum lateness.

Motivated by these practical and theoretical considerations, our primary objective in this dissertation is to devise scalable, efficient, and effective cut generation based algorithms for unrelated parallel machine scheduling problems. To this end, we study four  $\mathcal{NP}$ -hard unrelated parallel machine scheduling problems, two of which are proven to be  $\mathcal{NP}$ -hard in the strong sense. More specifically, the performance measures considered in this dissertation are total weighted tardiness (TWT), total weighted earliness/tardiness (TWET), total weighted completion time (TWCT), and last but not least, unrestrictive common due date total weighted earliness/tardiness (UCDD).

The specific motivations behind the selection of each performance measure and their significance are outlined in the introduction section of their respective chapters. Nevertheless, the common thread to all these performance measures is that even though they all are very fundamental scheduling objectives and studied extensively under other machine scheduling settings, they are not well studied in the unrelated parallel machine environment.

## 1.1 Outline

To facilitate the possibility of studying each problem independently, the results of this dissertation are presented in three separate main chapters. Each chapter is designed to be self contained in the sense that they can be read with little to no smattering from other chapters. In each chapter, we first present a brief summary of the work carried out, then introduce the problem and the motivation behind studying this specific performance measure. This is followed by the sections in which we present our main theoretical and methodological contributions, and we conclude each chapter with the results of the computational experiments.

In Chapter 2, we study TWT and TWET, and develop a new preemptive relaxation which provides tight lower bounds and near-optimal job to machine partitions. This relaxation turns out to be a hard to solve MIP problem and we devise a computationally effective Benders decomposition algorithm which can handle very large instances of this formulation. This chapter has been published as (Şen and Bülbül, 2015b).

Chapter 3 is dedicated to one of the most frequently studied fundamental scheduling objectives – i.e., TWCT. We prove that the preemptive relaxation of Chapter 2 is an exact formulation when the performance measure is TWCT. By exploiting the structural properties of TWCT, we attain a very fast and scalable exact Benders decomposition-based algorithm for solving this formulation. This chapter has been submitted as (Bülbül and Şen, 2015).

In Chapter 4, we consider the unrestricted common due date just-in-time scheduling problem and devise an exact LBB algorithm by studying the combinatorial structure of UCDD. The proposed solution approach turns out to be very efficient, and it is by far the best performing exact algorithm up to date for solving this hard scheduling problem. The manuscript of this chapter is in preparation (Şen and Bülbül, 2015a).

We conclude the dissertation in Chapter 5 with a summary of the conclusions drawn from Chapters 2–4 and indicate possible future research directions.



## CHAPTER 2

# A STRONG PREEMPTIVE RELAXATION FOR TOTAL WEIGHTED TARDINESS AND EARLINESS/TARDINESS

Research on due date oriented objectives in the parallel machine environment is at best scarce compared to objectives such as minimizing the makespan or the completion time related performance measures. Moreover, almost all existing work in this area is focused on the identical parallel machine environment. In this chapter, we leverage on our previous work on the single-machine total weighted tardiness (TWT) and total weighted earliness/tardiness (TWET) problems and develop a new preemptive relaxation for the TWT and TWET problems on a bank of unrelated parallel machines. The key contribution of this study is devising a computationally effective Benders decomposition algorithm for solving the preemptive relaxation formulated as a mixed integer linear programming (MIP) problem. The optimal solution of the preemptive relaxation provides a tight lower bound. Moreover, it offers a near-optimal partition of the jobs to the machines, and then we exploit recent advances in solving the non-preemptive single-machine TWT and TWET problems for constructing non-preemptive solutions of high quality to the original problem. We demonstrate the effectiveness of our approach with instances up to 5 machines and 200 jobs.

## 2.1 Introduction

Most of the studies in the scheduling literature are typically concerned with developing algorithms for a single objective function. The proposed approaches tend to be highly specialized and not easily extensible to other objectives and settings. Ultimately, scheduling software is tailored to individual settings, and scheduling research is fragmented. In this context, we emphasize that in this chapter we attack two popular scheduling objectives TWT and TWET within a single algorithmic framework. The TWT objective is a special case of the TWET objective; however, observe that TWET is non-regular while TWT is regular. It is well-established that non-regular objectives give rise to new theoretical and computational issues (Baker and Scudder, 1990, Kanet and Sridharan, 2000), and we point out that it is uncommon to tackle both objectives simultaneously. Formally, we characterize the problems we consider as  $Rm // \sum_j \pi_j T_j$  ( $Rm$ -TWT) and  $Rm // \sum_j \epsilon_j E_j + \pi_j T_j$  ( $Rm$ -TWET) for minimizing the TWT and TWET on a set of  $m$  unrelated parallel machines, respectively, following the three field notation of Graham et al. (1979) in classifying scheduling problems. The notation  $Rm$  in the first field stands for a bank of  $m$  unrelated machines. The earliness and tardiness of job  $j$  are represented by  $E_j$  and  $T_j$ , respectively, and  $\epsilon_j$  and  $\pi_j$  are the associated unit weights. Both  $Rm$ -TWT and  $Rm$ -TWET are strongly  $\mathcal{NP}$ -hard because the strongly  $\mathcal{NP}$ -hard single-machine scheduling problem  $1 // \sum \pi_j T_j$  (Lenstra et al., 1977) is a special case of both of these problems. We next summarize briefly our motivation and main contributions in this chapter.

The review of the related literature in Section 2.2 identifies the lack of strong lower bounds as a major impediment to the development of exact algorithms and the performance analysis of heuristics for the TWT and TWET objectives in the parallel machine environment. Shim and Kim (2007a) attack the unweighted version of  $Rm$ -TWT, and their branch-and-bound (B&B) algorithm does not scale beyond 5 machines and 20 jobs. In their concluding remarks, the authors state that “..., further research is needed if one needs to solve problems of larger or practical sizes. One way may be to develop more effective or tighter lower bounds since the lower bound used in the B&B algorithm suggested in this study does not seem to be very tight.” More generally, in their effort to compute

strong linear programming (LP) based bounds for a class of parallel machine scheduling problems with additive objectives, [van den Akker et al. \(1999\)](#) observe that “additive objective functions pose a computational challenge because it is difficult to compute strong lower bounds.” These comments provide a strong motivation for the study in this chapter. All promising existing results assume that the machines are identical and often exploit this fact in some way; e.g., by aggregating the machine capacity constraints. Clearly, such approaches do not necessarily extend to or yield similar results for unrelated parallel machines. In this chapter, we set out to provide tight lower bounds and near-optimal solutions for the TWT and TWET objectives in the unrelated parallel machine environment. To this end, we propose a new preemptive relaxation that explicitly assigns jobs to specific machines. This preemptive relaxation generalizes and builds upon the success of the related previous studies on the single-machine weighted tardiness and weighted earliness/tardiness scheduling problems ([Bülbül et al., 2007](#), [Pan and Shi, 2007](#), [Şen and Bülbül, 2012](#), [Sourd and Kedad-Sidhoum, 2003](#)). The resulting lower bound is tight, and perhaps more importantly, the job partition retrieved from the (near-)optimal solution of the preemptive relaxation provides us with sufficient information to construct feasible non-preemptive schedules of high quality for the original problem. That is, we recognize that the main practical difficulty of solving  $Rm$ -TWT and  $Rm$ -TWET to (near-)optimality is determining a good job partition, and we directly incorporate this aspect of the problem into our rationale for developing this particular relaxation. Once a job partition is available, we rely on recent advances by [Tanaka et al. \(2009\)](#) and [Tanaka and Fujikuma \(2012\)](#) to solve  $m$  independent single-machine TWT or TWET problems, respectively, to construct a non-preemptive solution of high quality to the original unrelated parallel machine scheduling problem. The downside of our preemptive relaxation is that it is formulated as a difficult MIP problem. A key contribution of this chapter is devising a computationally effective Benders decomposition algorithm that can handle very large instances of this formulation. Here, the *lazy constraint* generation scheme of [IBMILOG CPLEX \(2011\)](#) proves instrumental for a successful implementation. Moreover, as we point out in the previous paragraph, both objectives TWT and TWET are tackled successfully by the same algorithm.

In the next section, we review the related literature and put our work into

perspective. We introduce and formulate the proposed preemptive relaxation in Section 2.3 and then develop our solution approach based on Benders decomposition in Section 2.4. This is followed in Section 2.5 by an extensive set of computational experiments.

## 2.2 Review of Related Literature

Early research on parallel machine scheduling is primarily concerned with the makespan and total (weighted) completion time objectives (Cheng and Sin, 1990). We refer the reader to Pinedo (2008) for a comprehensive discussion of the polynomially solvable cases and structural results of interest for these problems. Some of the more recent and well-known examples of the papers that study  $\mathcal{NP}$ -complete problems in this domain include van den Akker et al. (1999), Chen and Powell (1999b), Azizoglu and Kirca (1999a), and Azizoglu and Kirca (1999b). Studies on due date related performance measures in the parallel machine environment commenced in earnest in the 1990's and picked up more significantly during the last decade. In this review, we mainly restrict our attention to the literature on parallel machine tardiness and earliness/tardiness scheduling problems with job dependent due dates. This part of the literature creates the context for our study, and we provide a few important pointers otherwise. The great majority of the existing studies on due date related performance measures assumes that the machines are identical, and only a handful of papers consider the case of unrelated parallel machines. For most of the proposed exact approaches, computational scalability remains an issue due to the lack of strong lower bounds. Therefore, we also specifically elaborate on the existing lower bounding methods for parallel machine scheduling problems with additive tardiness and earliness/tardiness objective functions in order to justify our alternate lower bounding scheme introduced in Section 2.3. See Table 2.1 at the end of this section for a summary of the important points in this section. Note that the performance figures presented in this section are obtained by their respective authors on different computing platforms.

The first exact approach for minimizing the total tardiness with distinct due dates on identical parallel machines is due to Azizoglu and Kirca (1998). The

authors integrate some dominance rules and a simple bounding technique into a B&B procedure for this problem  $Pm//\sum_j T_j$ , where  $Pm$  in the first field indicates a set of  $m$  identical parallel machines. The algorithm is able to handle instances with up to 15 jobs and 3 machines. The lower bound of [Azizoglu and Kirca](#) belongs to a very common and simple set of lower bounds which rely on determining a lower bound for the  $j$ th smallest job completion time  $C_{[j]}$ ,  $j = 1, \dots, n$ , among the set of all feasible schedules. These lower bounds on the completion times are then matched with the weights and the due dates in some appropriate order so that the resulting expression yields a lower bound for the problem under consideration. Lower bounding techniques based on such *minimal completion times* are developed or employed in several other papers with tardiness related objectives ([Koulamas, 1997](#), [Liaw et al., 2003](#), [Shim and Kim, 2007a,b](#), [Souayah et al., 2009](#), [Yalaoui and Chu, 2002](#)). There is a consensus in the literature that this class of lower bounds is not strong in general. Furthermore, in problems with earliness/tardiness objectives the presence of unforced idle time renders similar lower bounding techniques invalid. For the same problem  $Pm//\sum_j T_j$ , [Yalaoui and Chu \(2002\)](#) devise another B&B scheme. The limit of this algorithm appears to be 20 jobs and 2 machines within a time limit of 30 minutes.

The series of papers by [Liaw et al. \(2003\)](#), [Shim and Kim \(2007a\)](#), and [Shim and Kim \(2007b\)](#) develop a set of closely related optimal methods. [Liaw et al. \(2003\)](#) attack the problem  $Rm//\sum_j \pi_j T_j$  of minimizing TWT on unrelated parallel machines. This study appears to be the first exact approach for this problem. The lower bounding scheme is very similar to that in [Azizoglu and Kirca \(1999b\)](#) with a simple enhancement based on the structure of the tardiness objective; however, the method does not scale beyond 4 machines and 18 jobs. [Shim and Kim \(2007a\)](#) tackle the unweighted version  $Rm//\sum_j T_j$  in the same machine environment. The proposed B&B method employs some of the existing dominance properties in addition to new ones. The lower bounding technique of [Liaw et al. \(2003\)](#) is adopted, and an alternate lower bound is obtained by reducing the original problem into a single-machine problem by modifying the processing times appropriately and using a previously existing result for the single-machine total tardiness problem. The largest problem size that can be handled successfully within 1 hour is 5 machines and 20 jobs. In a similar work, [Shim and Kim \(2007b\)](#) address the problem

$Pm//\sum T_j$ , and instances with up to 5 machines and 30 jobs are solved optimally within 1 hour. [Jouglet and Savourey \(2011\)](#) devise dominance rules and filtering methods for the problem  $Pm/r_j/\sum_j \pi_j T_j$ , where the notation  $r_j$  in the second field indicates that the release dates may be non-identical, and embed these into a B&B procedure along with an existing lower bound. The authors argue that the lack of good lower bounds prevents them from solving instances with more than 20 jobs and 3 machines.

All of the optimal methods discussed so far base their lower bounding efforts on combinatorial arguments that rely on simple properties of the scheduling objectives under consideration. The resulting bounds are generally loose. However, the most promising lower bounds for parallel machine total (weighted) tardiness and earliness/tardiness problems are derived through mathematical programming techniques. For instance, the LP relaxations of the set partitioning formulations of common due date / common due window earliness/tardiness problems solved by column generation (CG) yield a prominent class of tight lower bounds ([Chen and Lee, 2002](#), [Chen and Powell, 1999a](#)). Bounds obtained from various relaxations of time-indexed formulations are also popular in parallel machine scheduling. An arc-time-indexed formulation whose LP relaxation is tackled by CG is at the heart of the highly efficient branch-cut-and-price algorithm of [Pessoa et al. \(2010\)](#) for  $Pm//\sum \pi_j T_j$ . This study is by far the most successful exact algorithm to date on parallel machine tardiness problems and delivers optimal solutions to instances with up to 100 jobs and 4 machines. [Tanaka and Araki \(2008\)](#) apply Lagrangian relaxation (LR) to the time-indexed formulation of the problem  $Pm//\sum T_j$  in an effort to develop tight lower bounds. Instances with up to 25 jobs and 10 machines are solved optimally. The average gap of the initial lower bound is 2.4% for the instances not solved at the root node. [Souayah et al. \(2009\)](#) take on the weighted version of the problem and study  $Pm//\sum_j \pi_j T_j$ . With a mix of combinatorial, mathematical programming, and LR based lower bounds, about half of the instances with up to 35 jobs and 2 machines are solved to optimality within 20 minutes. We refer the interested reader to the review paper [Sen et al. \(2003\)](#) where the tardiness literature on multi-machine systems is briefly addressed as well.

Following this discussion, two observations are due regarding the state of the

literature. First, there is a clear need for studying the tardiness related objectives in the unrelated parallel machine environment; we can pinpoint only two studies which focus on the unrelated parallel machine environment. Second, more than 20 to 30 jobs and a few machines seems to be generally beyond the reach for the existing exact methods, attributed to the lack of strong lower bounds. We hope to provide a potential remedy to this issue in this chapter.

Several heuristics have been proposed for minimizing the total (weighted) tardiness on identical parallel machines. Many of them apply list scheduling based on some priority index and sometimes enhance the initial schedule by local search. [Yalaoui and Chu \(2002\)](#) review several heuristics of this kind. An interesting deviation from the mainstream here is the decomposition heuristic by [Koulamas \(1997\)](#). The author heuristically extends the well-known decomposition principle valid for  $1//\sum T_j$  to the problem  $Pm//\sum T_j$  with very good results. At each iteration, the position of one job in the overall schedule is fixed, where the subproblems in the decomposition are solved by a fast and effective heuristic for  $Pm//\sum T_j$  that observes the decomposition principle for the individual machine schedules. Furthermore, a hybrid simulated annealing heuristic is devised which is outperformed by the decomposition heuristic based on the solution quality and time trade-off. The results for 100-job instances indicate that the proposed heuristics are on average about 10-11% away from optimality with respect to a lower bound. A recent list scheduling heuristic by [Biskup et al. \(2008\)](#) for  $Pm//\sum T_j$  yields somewhat better results than those of [Koulamas](#) for large instances with up to 5 machines and 200 jobs. An absolute assessment of the solution quality is not available due to the lack of a good lower bound or a scalable exact method. For the weighted version, i.e., the problem  $Pm//\sum \pi_j T_j$ , [Armentano and Yamashita \(2000\)](#) design a tabu search heuristic. For evaluation purposes, they benchmark their feasible solutions against the LR based lower bound by [Luh et al. \(1990\)](#). This lower bound is obtained by dualizing the machine capacity constraints in an integer programming formulation of the problem, similar to that by [Tanaka and Araki \(2008\)](#) discussed in the main text. In the original paper, [Luh et al.](#) include very limited computational experience, but the results of [Armentano and Yamashita \(2000\)](#) for instances with up to 10 machines and 150 jobs are promising. For instances with 100 jobs, the average optimality gap with respect to the LR

bound of [Luh et al. \(1990\)](#) is 8.14% which drops to 5.80% for 150-job instances. On the flip side, [Armentano and Yamashita](#) report that computing the lower bound of [Luh et al.](#) takes about 3 hours for 100- and 150-job instances.

For unrelated parallel machines, we are aware of only three papers by [Zhou et al. \(2007\)](#), [Mönch \(2008\)](#), and [Lin et al. \(2011\)](#) which focus on heuristics for  $Rm // \sum_j \pi_j T_j$ . The first two studies rely on ant colony optimization and benchmark their algorithms against simple heuristics which makes it difficult to evaluate the solution quality in absolute terms. [Lin et al.](#) propose a genetic algorithm and two simpler heuristics. The genetic algorithm outperforms all others in the computational experiments and deviates from the optimal solution by 1.8% on average for small instances with 4 machines and 20 jobs. The heuristic that we develop in this chapter is scalable to large instances with up to 200 jobs and simultaneously produces both lower and upper bounds of high quality. As evident from the discussion here, this is a significant edge over those in the literature, and we make a valuable contribution to the (unrelated) parallel machine scheduling research with tardiness objectives.

To the best of our knowledge, no exact algorithm has been designed to date for the problem of scheduling a set of independent jobs on a bank of unrelated parallel machines with the objective of minimizing the total (weighted) earliness and tardiness. However, various studies investigate special cases of this problem – see the literature review given in Section 4.2.1 of Chapter 4. The most closely related works to our problem  $Rm-TWET$  are by [Kedad-Sidhoum et al. \(2008\)](#), [Mason et al. \(2009\)](#), and [M’Hallah and Al-Khamis \(2012\)](#). [Kedad-Sidhoum et al.](#) experiment with various relaxations of the problem  $Pm/r_j / \sum_j \pi_j T_j + \epsilon_j E_j$  by recognizing that the main difficulty in solving earliness/tardiness scheduling problems stems from the lack of strong lower bounds. The authors extend two classes of lower bounds originally proposed for the single-machine case to the identical parallel machine environment. Their discrete assignment-based lower bound is discussed further in Section 2.3 because it is closely related to our preemptive lower bounding method for  $Rm-TWT$  and  $Rm-TWET$ . [Kedad-Sidhoum et al.](#) report that the LR obtained by dualizing the machine capacity constraints in the time-indexed formulation outperforms others, taking into account both the solution quality and gap. [Tanaka and Araki \(2008\)](#) – discussed previously – employ the same LR for



$Pm//\sum T_j$ . The best bound attained by solving the Lagrangian dual problem in these relaxations is equivalent to that provided by the LP relaxation of the time-indexed formulation. However, solving the Lagrangian dual problem – generally by subgradient optimization – is often computationally more efficient. We also attest to the rapidly increasing computational effort required to solve the LP relaxation of the time-indexed formulation in Section 2.5. [Kedad-Sidhoum et al.](#) obtain upper bounds through a simple local search. Experimental results attest to the quality of both the lower and upper bounds. The average optimality gap attained for instances with up to six machines and 90 jobs is around 1.5%. However, we cite two good reasons for not following a similar path to that of [Kedad-Sidhoum et al.](#) and [Tanaka and Araki](#). First, the machine capacity constraints in the time-indexed formulation may be aggregated in the identical parallel machine environment by defining a single resource with a capacity of performing  $m$  jobs simultaneously, and this renders the number of dual variables in the LR independent from the number of machines in the problem. This, however, is not possible for  $Rm-TWT$  and  $Rm-TWET$ , and relaxing the machine capacity constraints – one for each combination of time period and machine – would result in  $mH$  dual variables instead of just  $H$ . Consequently, solving the Lagrangian dual problem would quickly become a formidable task with an increasing number of machines. Second, the solution retrieved from the LR does offer little information on how to identify near-optimal job to machine assignments. The job start times provided by the LR for a given set of dual multipliers form the basis for a dispatch rule in [Tanaka and Araki \(2008\)](#); however, both these authors and [Kedad-Sidhoum et al.](#) need to devise independent heuristics in order to obtain feasible solutions of high-quality for their original problems.

The moving block heuristic of [Mason et al. \(2009\)](#) for  $Pm//\sum E_j + T_j$  is tested against an integer programming formulation over instances with up to 40 jobs and 4 machines. The heuristic identifies feasible solutions which are on average better than the incumbent for 20- and 40-job instances. Like [Kedad-Sidhoum et al.](#), [M'Hallah and Al-Khamis](#) tackle the weighted version of the problem. Their integer programming formulation points out and corrects an error in that of [Mason et al. \(2009\)](#). The limit of the formulation appears to be instances with no more than 20 jobs. In addition, several new heuristics are introduced. The best

performing contender turns out to be a hybrid heuristic which is benchmarked against the lower and upper bounds of [Kedad-Sidhoum et al. \(2008\)](#). The hybrid heuristic improves some of the best known solutions for the instances of [Kedad-Sidhoum et al.](#); however, it yields slightly worse solutions on average. The median gap of the hybrid heuristic ranges from 1.4% to 6.1% with respect to the lower bounds of [Kedad-Sidhoum et al. \(2008\)](#) depending on the problem size. It is evident that there is a gap in the literature with respect to the parallel machine earliness/tardiness scheduling problems with distinct due dates. To the best of our knowledge, our work provides the first viable solution approach for the unrelated parallel machine environment in this context.

**Table 2.1** Summary of the important points in Section 2.2.

Paper	Problem	Method	Main Results. $[n, m]^{\dagger}$ , Time/Gap <sup>†</sup>
<a href="#">Liaw et al. (2003)</a>	$Rm // \sum_j \pi_j T_j$	Exact	B&B. First exact approach. [18, 4]
<a href="#">Shim and Kim (2007a)</a>	$Rm // \sum_j T_j$	Exact	B&B. Bound of <a href="#">Liaw et al. (2003)</a> and an alternate one. [20, 5], 60 min
<a href="#">Zhou et al. (2007)</a>	$Rm // \sum_j \pi_j T_j$	Heuristic	Ant colony optimization
<a href="#">Mönch (2008)</a>	$Rm // \sum_j \pi_j T_j$	Heuristic	Ant colony optimization. ATC dispatching, decomposition heuristic
<a href="#">Lin et al. (2011)</a>	$Rm // \sum_j \pi_j T_j$	Heuristic	Genetic algorithm and two simple heuristics. [20, 4], 1.8%
<a href="#">Plateau and Rios-Solis (2010)</a>	$Rm/d_j = d/\sum_j \epsilon_j E_j + \pi_j T_j$	Exact	Convex quadratic integer programming (CQIP) formulation. Results for $d_j = d^r$ not satisfactory. [50, 4], 60 min
<a href="#">Azizoglu and Kirca (1998)</a>	$Pm // \sum_j T_j$	Exact	B&B. Dominance rules. Lower bound based on <i>minimal completion times</i> . [15, 3]
<a href="#">Yalaoui and Chu (2002)</a>	$Pm // \sum_j T_j$	Exact	B&B. [20, 2], 30 min
<a href="#">Shim and Kim (2007b)</a>	$Pm // \sum_j T_j$	Exact	B&B. Dominance rules. [30, 5], 60 min
<a href="#">Tanaka and Araki (2008)</a>	$Pm // \sum_j T_j$	Exact	Lagrangian relaxation (LR) to time-indexed formulation. [25, 10]
<a href="#">Souayah et al. (2009)</a>	$Pm // \sum_j \pi_j T_j$	Exact	Mix of bounds. [35, 2], 20 min
<a href="#">Pessoa et al. (2010)</a>	$Pm // \sum_j \pi_j T_j$	Exact	Branch-cut-and-price. Arc-time-indexed. Best to date. [100, 4]
<a href="#">Jouglet and Savourey (2011)</a>	$Pm/r_j // \sum_j \pi_j T_j$	Exact	B&B. Dominance rules. [20, 3]
<a href="#">Koulamas (1997)</a>	$Pm // \sum_j T_j$	Heuristic	Decomposition heuristic. [100, 8], ~ 10%
<a href="#">Armentano and Yamashita (2000)</a>	$Pm // \sum_j \pi_j T_j$	Heuristic	Tabu search. LR of <a href="#">Luh et al. (1990)</a> . [150, 10], 5%-10%
<a href="#">Biskup et al. (2008)</a>	$Pm // \sum_j T_j$	Heuristic	List scheduling. [200, 5]
<a href="#">Chen and Powell (1999a)</a>	$Pm/d_j = d^l // \sum_j \epsilon_j E_j + \pi_j T_j$	Exact	Set partitioning formulation. Dantzig-Wolfe decomposition. [60, 6]
<a href="#">Chen and Lee (2002)</a>	$Pm/[d_1, d_2] // \sum_j \epsilon_j E_j + \pi_j T_j$	Exact	Extends ( <a href="#">Chen and Powell, 1999a</a> ). Common due window. Column generation (CG). [40, $m$ ]
<a href="#">Kedad-Sidhoum et al. (2008)</a>	$Pm/r_j // \sum_j \epsilon_j E_j + \pi_j T_j$	Heuristic	LR to time-indexed formulation. [90, 6], 1.5%
<a href="#">Rios-Solis and Sourd (2008)</a>	$Pm/d_j = d^r // \sum_j \epsilon_j E_j + \pi_j T_j$	Heuristic	Dynamic programming to explore exponential-size neighborhood

<sup>†</sup>: Largest instance size tackled successfully and the associated time / optimality gap information if available.

Continued on next page...

Table 2.1 continued...

Paper	Problem	Method	Main Results. $[n, m]^{\dagger}$ , Time/Gap <sup>†</sup>
<a href="#">Mason et al. (2009)</a>	$Pm // \sum_j E_j + T_j$	Heuristic	Moving-block heuristic. [40, 4]
<a href="#">M'Hallah and Al-Khamis (2012)</a>	$Pm // \sum_j \epsilon_j E_j + \pi_j T_j$	Heuristic	Two constructive and three meta-heuristics. [90, 6], 1.4%-6.4%
<a href="#">Cheng and Sin (1990)</a>			"A State-of-the-Art Review of Parallel-Machine Scheduling"
<a href="#">Baker and Scudder (1990)</a>			"Sequencing with Earliness and Tardiness Penalties: A Review"
<a href="#">Kanet and Sridharan (2000)</a>			"Scheduling with Inserted Idle Time: Problem Taxonomy and Literature Review"
<a href="#">Sen et al. (2003)</a>			"Static Scheduling Research to Minimize Weighted and Unweighted Tardiness: A State-of-the-Art Survey"
<a href="#">Lauff and Werner (2004)</a>			"Scheduling with Common Due Date, Earliness and Tardiness Penalties for Multimachine Problems: A Survey"

<sup>†</sup>: Largest instance size tackled successfully and the associated time / optimality gap information if available.

### 2.3 Problem Statement and Preemptive Relaxation

We consider a bank of  $m$  unrelated parallel machines and  $n$  jobs, which are all ready at time zero. Each job is processed on exactly one of the machines, where the processing of job  $j$  on machine  $k$  requires an integer duration of  $p_{jk}$  time units. The completion time of job  $j$  is denoted by  $C_j$ . A due date  $d_j$  – also assumed to be integral – is associated with each job  $j$ , and we incur a cost  $\pi_j$  per unit time if job  $j$  completes processing after  $d_j$ . Thus, the total weighted tardiness over all jobs is determined as  $\sum_j \pi_j T_j$ , where the tardiness of job  $j$  is calculated as  $T_j = \max(0, C_j - d_j)$ . For the problem  $Rm // \sum_j \pi_j T_j + \epsilon_j E_j$ , the objective additionally penalizes the completion of job  $j$  prior to its due date  $d_j$  at a rate of  $\epsilon_j$  per unit time, where the earliness of job  $j$  is defined as  $E_j = \max(0, d_j - C_j)$ . All machines are available continuously from time zero onward, and a machine can execute at most one operation at a time. An operation must be carried out to completion once started, i.e., preemption is not allowed.

In this section, we introduce our preemptive lower bounding scheme for  $Rm$ - $TWT$  and  $Rm$ - $TWET$ . We define two primary design goals for our preemptive relaxation. The tightness of the lower bound is clearly a major concern. Equally important is the information that can be extracted from the optimal solution of the preemptive relaxation to construct feasible solutions of high quality for the original non-preemptive problem. We attain both of these goals – somewhat more successfully for  $Rm$ - $TWT$  than for  $Rm$ - $TWET$  from a computational perspective – and demonstrate the effectiveness of the proposed lower and upper bounds in Section 2.5.

A class of highly efficient lower bounds based on a particular preemption scheme was developed for single-machine tardiness and earliness/tardiness scheduling problems during the last decade (Bülbül et al., 2007, Şen and Bülbül, 2012, Sourd and Kedad-Sidhoum, 2003). The key idea of these preemptive relaxations is to divide up jobs with integer processing times into jobs of unit-length and associate a cost with the completion of each of these unit-length jobs. That is, jobs may only be preempted at integer points in time. In this setting, the problem of solving the preemptive relaxation is formulated as an assignment or a transportation problem, where the length of the planning horizon depends on the

magnitude of the due dates and the sum of the processing times. Therefore, the formulation size is pseudo-polynomial. On the up side, the availability of very fast algorithms for the assignment and transportation problems does still render this lower bounding technique viable. The formulation **(TR)** below is due to [Kedad-Sidhoum et al. \(2008\)](#), where the original approach in the single-machine environment is extended to  $m$  identical parallel machines.

$$\text{(TR)} \quad \text{minimize} \quad \sum_{j=1}^n \sum_{t=1}^H c'_{jt} x_{jt} \quad (2.1)$$

$$\text{subject to} \quad \sum_{t=1}^H x_{jt} = p_j, \quad j = 1, \dots, n, \quad (2.2)$$

$$\sum_{j=1}^n x_{jt} \leq m, \quad t = 1, \dots, H, \quad (2.3)$$

$$0 \leq x_{jt} \leq 1, \quad j = 1, \dots, n, t = 1, \dots, H. \quad (2.4)$$

In the model **(TR)**, the time period  $t$  represents the time interval  $(t - 1, t]$ , and consequently in any optimal schedule all jobs finish processing no later than in period  $H$ , where

$$H = \begin{cases} \left\lceil \left[ \sum_{j=1}^n \max_k(p_{jk}) / m \right] + p_{\max} \right\rceil & \text{for } Rm\text{-TWT, and} \\ \left\lceil \left[ \sum_{j=1}^n \max_k(p_{jk}) / m \right] + p_{\max} + d_{\max} \right\rceil & \text{for } Rm\text{-TWET.} \end{cases} \quad (2.5)$$

The end of the planning horizon  $H$  is determined based on the following observation. For  $Rm\text{-TWT}$  with a regular objective function, all machines are continuously busy until some time  $t' \leq \left\lceil \sum_{j=1}^n \max_k(p_{jk}) / m \right\rceil$  if at least  $m$  jobs are still not completed. Therefore, after time  $t'$  the remaining  $m - 1$  jobs are finished in at most  $p_{\max} = \max_{j,k}(p_{jk})$  time periods. The end of the planning horizon may thus be set to the value in the first row of (2.5). An optimal solution of  $Rm\text{-TWET}$ , on the other hand, may include unforced idleness, and the argument just described is only valid if we conservatively assume that all jobs are started at  $d_{\max} = \max_j d_j$ . Clearly,  $p_{\max}$  may be omitted from (2.5) in the case of a single-machine.

If a unit job of job  $j$  is executed during the time interval  $(t - 1, t]$ , the decision

variable  $x_{jt}$  assumes the value one, and the objective is charged a cost of  $c'_{jt}$ . The constraints (2.2) mandate that each job  $j$  receives  $p_j$  units of processing. To observe the machine capacities, constraints (2.3) require that no more than  $m$  unit jobs are processed simultaneously in a given period. Note that the machine index is omitted from the processing times because they are all identical for a given job. Furthermore, no integrality is imposed on the decision variables due to the total unimodularity of the constraint matrix of (TR). The optimal objective function value of (TR) is a lower bound on that of  $Pm // \sum_j \pi_j T_j + \epsilon_j E_j$ , as long as the objective function coefficients satisfy

$$\sum_{s=t-p_j+1}^t c'_{js} \leq \epsilon_j (d_j - t)^+ + \pi_j (t - d_j)^+, \quad j = 1, \dots, n, \quad t = p_j, \dots, H. \quad (2.6)$$

That is, the total cost incurred in (TR) by any job that is scheduled non-preemptively is no larger than that in the original non-preemptive problem (Bülbül et al., 2007). Naturally, the strength of the lower bound depends on the objective coefficients  $c'_{jt}$ , and this is where the existing works in the literature take different paths. For instance, the cost coefficients of Sourd and Kedad-Sidhoum (2003) satisfy (2.6) as an equality. Bülbül et al. (2007) characterize and develop an expression for the cost coefficients that are the best among those with a piecewise linear structure with two segments. For these cost coefficients, (2.6) holds as a strict inequality for some values of  $t$ . For the one machine problem, these authors also show that the lower bound retrieved from (TR) is no better than that provided by the LP relaxation of the time-indexed formulation. Conversely, Pan and Shi (2007) prove the existence of a set of objective coefficients for (TR) so that the LP relaxation of the time-indexed formulation and (TR) yield identical lower bounds. However, computing the values of these cost coefficients is no less time consuming than solving the LP relaxation of the time-indexed formulation. We also note that the empirical performance of the algorithms based on this set of relaxations is more than satisfactory (Bülbül et al., 2007, Pan and Shi, 2007, Şen and Bülbül, 2012, Sourd and Kedad-Sidhoum, 2003). They strike a good balance between solution quality and time.

Factoring in all arguments in this section, the set of preemptive relaxations

discussed in the previous paragraph emerges as a strong candidate for deriving strong lower bounds for our problems of interest  $Rm-TWT$  and  $Rm-TWET$ . However, one hurdle remains in the pursuit of our second design goal of constructing non-preemptive solutions of high quality directly based on the information retrieved from the optimal solution of the preemptive relaxation. In the optimal solution of **(TR)**, the unit jobs of job  $j$  cannot overlap in time, but they can be processed on different machines. Consequently, no explicit assignment of the jobs to the machines is available. This is a major drawback because it complicates the task of obtaining a non-preemptive feasible solution to the original problem. In the sequel, we demonstrate that overcoming this difficulty allows us to attain good upper bounds in addition to good lower bounds.

The downside of **(TR)** is that the optimal solution does not guarantee that we can assign all unit jobs of a job to the same machine. Such a requirement is incorporated in the following model **(TR – A)** at the expense of additional variables and destroying the desirable polyhedral structure of the transportation problem. The binary variable  $y_{jk}$  takes the value 1 if job  $j$  is assigned to machine  $k$ , and is zero otherwise. In addition, the  $x$ -variables and the associated objective coefficients are supplemented with a machine index  $k$  to allow us to assign a unit job of job  $j$  explicitly to machine  $k$  in period  $t$ .

$$\text{(TR – A) minimize } \sum_{j=1}^n \sum_{k=1}^m \sum_{t=1}^H c_{jkt} x_{jkt} \quad (2.7)$$

$$\text{subject to } \sum_{t=1}^H x_{jkt} = p_{jk} y_{jk}, \quad j = 1, \dots, n, k = 1, \dots, m, \quad (2.8)$$

$$\sum_{j=1}^n x_{jkt} \leq 1, \quad k = 1, \dots, m, t = 1, \dots, H, \quad (2.9)$$

$$\sum_{k=1}^m y_{jk} = 1, \quad j = 1, \dots, n, \quad (2.10)$$

$$x_{jkt} \geq 0, \quad j = 1, \dots, n, k = 1, \dots, m, t = 1, \dots, H, \quad (2.11)$$

$$y_{jk} \in \{0, 1\}, \quad j = 1, \dots, n, k = 1, \dots, m. \quad (2.12)$$

**(TR – A)** differs from **(TR)** in two main aspects. The capacity constraints (2.9) appear in a disaggregated form, and all unit jobs of job  $j$  are performed on the



same machine by constraints (2.8) and the job partitioning constraints (2.10). As we hinted at earlier, the cost coefficients  $c'_{jkt}$  are of critical importance for the strength of the lower bounds provided by the preemptive relaxation. In this research, we stick with the cost coefficients by Bülbül et al. (2007) given in (2.13) and adapted in an obvious way to the unrelated parallel machine environment for two reasons. They empirically outperform those by Sourd and Kedad-Sidhoum (2003) on average (Bülbül et al., 2007, Kedad-Sidhoum et al., 2008), and computing the best set of cost coefficients for a given instance by the method of Pan and Shi (2007) is expensive.

$$c'_{jkt} = \begin{cases} \frac{\epsilon_j}{p_{jk}} \left[ \left( d_j - \frac{p_{jk}}{2} \right) - \left( t - \frac{1}{2} \right) \right] & \text{for } t \leq d_j, \text{ and} \\ \frac{\pi_j}{p_{jk}} \left[ \left( t - \frac{1}{2} \right) - \left( d_j - \frac{p_{jk}}{2} \right) \right] & \text{for } t > d_j. \end{cases} \quad (2.13)$$

We next provide a proposition that the optimal solution of (TR – A) with the cost coefficients given above provides a lower bound on the optimal objective function value of the original problem *Rm-TWT* or *Rm-TWET*. The result is a corollary of Bülbül et al. (2007, Theorem 2.2), where the authors show that the cost coefficients in (2.13) satisfy (2.6).

**Proposition 2.1.** *The optimal objective function value of (TR – A) with the cost coefficients given by equation (2.13) is a lower bound on the optimal objective function value of the original non-preemptive problem *Rm-TWT* or *Rm-TWET*.*

*Proof.* Let  $S_p$  represent a feasible schedule for problem (P) with a total cost of  $TC(S_p)$ . The notation  $(P(\bar{y}))$  stands for problem (P) in which the jobs are assigned to the machines a priori, but the individual machine schedules for this job partition  $\bar{y}$  are to be optimized. An optimal schedule is denoted by an asterisk in the superscript.

For any given fixed job partition  $\bar{y}$ , both the original non-preemptive problems *Rm-TWT* and *Rm-TWET* – denoted by (NP) – and the preemptive relaxation decompose into  $m$  independent single-machine problems. Therefore, we have  $TC(S_{NP(\bar{y})}^*) = \sum_{k=1}^m TC(S_{NP(\bar{y}_k)}^*)$  and  $TC(S_{TR-A(\bar{y})}^*) = \sum_{k=1}^m TC(S_{TR-A(\bar{y}_k)}^*)$ , where  $S_{NP(\bar{y}_k)}^*$  and  $S_{TR-A(\bar{y}_k)}^*$  stand for the optimal non-preemptive and preemptive schedules on machine  $k$  under  $\bar{y}$ , respectively. By Bülbül et al. (2007, Theorem 2.2),

$TC\left(S_{\text{TR-A}}^*(\bar{\mathbf{y}}_k)\right) \leq TC\left(S_{\text{NP}}^*(\bar{\mathbf{y}}_k)\right)$  for  $k = 1, \dots, m$ , and we have

$$TC\left(S_{\text{TR-A}}^*(\bar{\mathbf{y}})\right) = \sum_{k=1}^m TC\left(S_{\text{TR-A}}^*(\bar{\mathbf{y}}_k)\right) \leq \sum_{k=1}^m TC\left(S_{\text{NP}}^*(\bar{\mathbf{y}}_k)\right) = TC\left(S_{\text{NP}}^*(\bar{\mathbf{y}})\right).$$

This relationship is independent from  $\bar{\mathbf{y}}$  and does also hold for the optimal job partition  $\bar{\mathbf{y}}^*$  which concludes the proof.  $\square$

Our overall strategy for obtaining near-optimal feasible solutions and good lower bounds for  $Rm$ -TWT and  $Rm$ -TWET is now clear. We first solve  $(\text{TR} - \mathbf{A})$ , retrieve the job partition, and then build  $m$  individual machine schedules independently. Several heuristics with excellent empirical performance are available for both  $1/\sum_j \pi_j T_j$  and  $1/\sum_j \pi_j T_j + \epsilon_j E_j$  to perform the latter task. However, in this work we rely on the recent powerful optimal algorithms of [Tanaka et al. \(2009\)](#) and [Tanaka and Fujikuma \(2012\)](#) to handle the single-machine problems as we mentioned in Section 2.1. Our computational experiments in Section 2.5 ultimately support this decision. Thus, only one major challenge remains. The formulation  $(\text{TR} - \mathbf{A})$  is an MIP problem that is time consuming to solve based on our preliminary computational experiments. However, for a fixed job partition it decomposes into  $m$  independent LP problems –  $m$  independent transportation problems –, and these LP problems are solved to optimality very efficiently. These observations suggest that  $(\text{TR} - \mathbf{A})$  is amenable to Benders decomposition ([Benders, 1962](#)), and developing a Benders decomposition algorithm with *strengthened cuts* for  $(\text{TR} - \mathbf{A})$  is our main methodological contribution in this chapter.

One final remark is due before we delve into the specifics of our solution method for  $(\text{TR} - \mathbf{A})$ . For  $Rm$ -TWT, the formulation may be strengthened by the load balancing constraints (2.14) which assert that the workloads of two machines cannot differ by more than  $p_{\max}$  in an optimal solution of the original non-preemptive parallel machine scheduling problem. Otherwise, we could transfer the final job on one of these machines to the other one without degrading the objective function value. Note that similar concepts have been incorporated into various properties and dominance rules elsewhere in the literature ([Azizoglu and Kirca, 1999b](#), Theorem 1). However, for  $Rm$ -TWET with a non-regular objective function,

we can easily create instances for which no optimal solution satisfies (2.14).

$$-p_{\max} \leq \sum_{j=1}^n p_{jk} y_{jk} - \sum_{j=1}^n p_{jl} y_{jl} \leq p_{\max}, \quad k = 1, \dots, m-1, l = k+1, \dots, m. \quad (2.14)$$

These cuts are added to the preemptive formulation (TR – A) when solving  $Rm$ -TWT and help speed up the solution process for large instances.

## 2.4 Benders Decomposition

Parallel machine scheduling problems have a partitioning and a scheduling component. That is, if we assign jobs to machines by fixing the variables  $y_{jk}$ ,  $j = 1, \dots, n, k = 1, \dots, m$ , so that the constraints (2.10) are satisfied, then the model (TR – A) decomposes into  $m$  independent transportation problems. We exploit this key observation to design an algorithm based on Benders decomposition for solving (TR – A) efficiently. To this end, we reformulate (TR – A) for a fixed  $\bar{\mathbf{y}}$  by replacing the right hand side of the set of constraints (2.8) by  $p_{jk}\bar{y}_{jk}$  and dropping the set of constraints (2.10) and (2.12) from the model. In the resulting LP problem (TR – A( $\bar{\mathbf{y}}$ )),  $u_{jk}$ ,  $j = 1, \dots, n$ ,  $k = 1, \dots, m$ , and  $v_{kt}$ ,  $k = 1, \dots, m$ ,  $t = 1, \dots, H$ , are the dual variables associated with the set of constraints (2.8) and (2.9), respectively. The dual of (TR – A( $\bar{\mathbf{y}}$ )) is then stated below, where the decomposition into  $m$  independent transportation problems is made explicit:

$$z(\bar{\mathbf{y}}) = \sum_{k=1}^m z_k(\bar{\mathbf{y}}), \quad (2.15)$$

where

$$(\mathbf{DS}_k - \mathbf{F}) \quad z_k(\bar{\mathbf{y}}) = \text{maximize} \quad \sum_{j=1}^n p_{jk}\bar{y}_{jk}u_{jk} + \sum_{t=1}^H v_{kt} \quad (2.16)$$

$$\text{subject to} \quad u_{jk} + v_{kt} \leq c_{jkt}, \quad j = 1, \dots, n, t = 1, \dots, H, \quad (2.17)$$

$$v_{kt} \leq 0, \quad t = 1, \dots, H, \quad (2.18)$$

is the dual of the transportation problem  $(\mathbf{TR}_k)$  for machine  $k$ . In the sequel,  $(\mathbf{TR}_k)$  and  $(\mathbf{DS}_k - \mathbf{F})$  are also referred to as the *cut generation subproblem* and the *dual slave problem*, respectively, by following the common terminology for Benders decomposition.

Based on the objective function (2.16) of  $(\mathbf{DS}_k - \mathbf{F})$ , we obtain the following relaxed Benders master problem  $(\mathbf{RMP})$ , where  $C$  denotes the current number of times the cut generation subproblems  $(\mathbf{TR}_k)$ ,  $k = 1, \dots, m$ , have been solved. The optimal values of the dual variables  $\bar{u}_{jk}$ ,  $j = 1, \dots, n, k = 1, \dots, m$ , and  $\bar{v}_{kt}$ ,  $k = 1, \dots, m, t = 1, \dots, H$ , in round  $c$  of the cut generation are represented by  $\bar{u}_{jk}^c$  and  $\bar{v}_{kt}^c$ , respectively. The auxiliary variable  $\eta_k$  indicates a lower bound on the total cost incurred by the jobs assigned to machine  $k$ , and the objective function value  $\sum_{k=1}^m \eta_k$  of  $(\mathbf{RMP})$  is therefore a lower bound on the optimal objective values of  $(\mathbf{TR} - \mathbf{A})$  and the original non-preemptive scheduling problem  $Rm\text{-TWT}$  or  $Rm\text{-TWET}$ .

$$(\mathbf{RMP}) \quad \text{minimize} \quad \sum_{k=1}^m \eta_k \quad (2.19)$$

$$\text{subject to} \quad \sum_{k=1}^m y_{jk} = 1, \quad j = 1, \dots, n, \quad (2.20)$$

$$\eta_k \geq \sum_{j=1}^n p_{jk} \bar{u}_{jk}^c y_{jk} + \sum_{t=1}^H \bar{v}_{kt}^c, \quad k = 1, \dots, m, \quad c = 1, \dots, C, \quad (2.21)$$

$$y_{jk} \in \{0, 1\}, \quad j = 1, \dots, n, \quad k = 1, \dots, m. \quad (2.22)$$

Note that  $(\mathbf{TR} - \mathbf{A}(\bar{\mathbf{y}}))$  is feasible and  $(\mathbf{DS}_k - \mathbf{F})$ ,  $k = 1, \dots, m$ , is bounded for any  $\bar{\mathbf{y}}$  that satisfies the constraints (2.10). Therefore, no feasibility cuts are required, and only optimality cuts are generated and added iteratively to  $(\mathbf{RMP})$  during the course of the algorithm. Furthermore, the cut generation subproblem  $(\mathbf{TR}_k)$  for machine  $k$  includes all jobs and is solved by considering the full length of the planning horizon. From a computational point of view, however, we are better off by defining the set of jobs to be processed on machine  $k$  as  $J_k = \{j \mid y_{jk} = 1\}$ , setting the last period of processing on machine  $k$  – designated by  $H_k$  – as appropriate based on (2.5), and then solving a restricted version of  $(\mathbf{TR}_k)$  over these jobs and time periods only. This restricted cut generation subproblem formulation and the corresponding dual slave problem are referred to as  $(\mathbf{TR}_k - \mathbf{R})$  and  $(\mathbf{DS}_k - \mathbf{R})$ ,

respectively. Obviously, the optimal solution of  $(\mathbf{TR}_k - \mathbf{R})$  may be extended to an optimal solution of  $(\mathbf{TR}_k)$  trivially by setting  $x_{jkt} = 0$  for  $j \in J_k$ ,  $t = H_k + 1, \dots, H$ , and  $j \notin J_k$ ,  $t = 1, \dots, H$ . The relationship between the optimal solutions of  $(\mathbf{DS}_k - \mathbf{F})$  and  $(\mathbf{DS}_k - \mathbf{R})$  and its implications for the dynamic generation of the constraints (2.21) require a deeper discussion which is relegated to the next section.

In the *multi-cut* relaxed master problem formulation (**RMP**) above, we approximate the objective function of  $(\mathbf{TR} - \mathbf{A})$  by estimating the cost accumulated on each machine separately as evident from the set of constraints (2.21). Alternatively, we could have employed a weaker *single-cut* version of the relaxed master problem by aggregating all  $m$  cuts generated after solving  $(\mathbf{TR}_k)$ ,  $k = 1, \dots, m$ , and replacing  $\sum_{k=1}^m \eta_k$  by a single variable  $\eta$  in the formulation as appropriate. The single-cut version results in fast solution times for the relaxed master problem at the expense of more iterations overall. Ultimately, the trade-off between these two alternatives is only decided during the computations. In our preliminary testing, the cut generation algorithm based on (**RMP**) was clearly superior to that based on the single-cut version in terms of speed. Thus, the rest of the chapter is focused exclusively on (**RMP**). The pseudo-code of the cut generation procedure is stated in Algorithm 1.

---

**Algorithm 1:** Procedure *generate\_cuts*.

---

**input** : A feasible partition  $\bar{\mathbf{y}}$  of jobs to machines.  
**output**: Returns  $z_k(\bar{\mathbf{y}})$  and the strengthened cuts of the form (2.24) for  $k = 1, \dots, m$ .

- 1 **for**  $k = 1$  to  $m$  **do**
- 2     Solve  $(\mathbf{TR}_k - \mathbf{R})$ , retrieve  $z_k(\bar{\mathbf{y}})$  and the optimal solution  $(\bar{\mathbf{u}}_k, \bar{\mathbf{v}}_k)$  for the dual slave  $(\mathbf{DS}_k - \mathbf{R})$ ;
- /\* Calculate an alternate optimal solution  $(\bar{\mathbf{u}}'_k, \bar{\mathbf{v}}'_k)$  for  $(\mathbf{DS}_k - \mathbf{R})$  that satisfies Lemma 2.2 by following the construction in the proof. \*/
- 3      $\bar{v}_k^{\max} = \max_{t=1, \dots, H_k} \bar{v}_{kt}$ ;
- 4     **if**  $\bar{v}_k^{\max} < 0$  **then**  $\bar{u}'_{jk} = \bar{u}_{jk} - |\bar{v}_k^{\max}|$ ,  $j \in J_k$ ,  $\bar{v}'_{kt} = \bar{v}_{kt} + |\bar{v}_k^{\max}|$ ,  $t = 1, \dots, H_k$  **else**  $(\bar{\mathbf{u}}'_k, \bar{\mathbf{v}}'_k) = (\bar{\mathbf{u}}_k, \bar{\mathbf{v}}_k)$ ;
- // Construct an optimal solution  $(\bar{\mathbf{u}}''_k, \bar{\mathbf{v}}''_k)$  for  $(\mathbf{DS}_k - \mathbf{F})$ .
- 5      $\bar{v}''_{kt} = \bar{v}'_{kt}$ ,  $t = 1, \dots, H_k$ , and  $\bar{v}''_{kt} = 0$ ,  $t = H_k + 1, \dots, H$ ;
- 6      $\bar{u}''_{jk} = \bar{u}'_{jk}$ ,  $j \in J_k$ , and  $\bar{u}''_{jk}$ ,  $j \notin J_k$ , is calculated based on either (2.29) or (2.30), respectively, depending on whether we solve an instance of *Rm-TWT* or *Rm-TWET*;
- 7     Generate and add (2.24) to *cuts*;

---

### 2.4.1 Validity and Strengthening of the Benders Cuts

The validity of Benders decomposition (Benders, 1962) derives from the independence of the feasible region of the dual slave problem from the values of the integer variables. For an MIP problem of the form minimize  $\{g\mathbf{x} + h\mathbf{y} : G\mathbf{x} + H\mathbf{y} \geq \mathbf{b}, \mathbf{x} \in \mathbf{R}^+, \mathbf{y} \in \mathbf{Z}^+\}$ , where all matrices and vectors have appropriate dimensions, the dual slave problem for a given  $\bar{\mathbf{y}}$  is stated as maximize  $\{w^T(\mathbf{b} - H\bar{\mathbf{y}}) : w^T G \leq \mathbf{g}, w \in \mathbf{R}^+\}$ , where  $w$  is the vector of dual variables of appropriate size. In other words, the dual slave problem is always solved over the same dual polyhedron  $\{w^T G \leq \mathbf{g}, w \in \mathbf{R}^+\}$ , and only the objective function depends on the values of the integer variables. As a consequence, the maximum number of cuts to be generated is bounded from above by the number of extreme points of the dual polyhedron. These issues need a closer look, however, if we opt for solving  $(\mathbf{TR}_k - \mathbf{R})$  instead of  $(\mathbf{TR}_k)$  because this amounts to solving the dual slave problem over different feasible regions every time and contradicts the basic pillar of Benders decomposition. Observe that a cut of the form

$$\eta_k \geq \sum_{j \in J_k} p_{jk} \bar{u}_{jk} y_{jk} + \sum_{t=1}^{H_k} \bar{v}_{kt} \quad (2.23)$$

produced directly out of an optimal solution of  $(\mathbf{DS}_k - \mathbf{R})$  relies on the assumption that augmenting this solution trivially with  $\bar{u}_{jk} = 0$  for  $j \notin J_k$  and  $\bar{v}_{kt} = 0$  for  $t = H_k + 1, \dots, H$ , is feasible with respect to  $(\mathbf{DS}_k - \mathbf{F})$ . It is a simple matter to show that as long as the optimal solution of  $(\mathbf{DS}_k - \mathbf{R})$  satisfies  $\max_{t=1, \dots, H_k} \bar{v}_{kt} = 0$  (see Lemma 2.2), this augmented solution is feasible with respect to  $(\mathbf{DS}_k - \mathbf{F})$  if we are solving an instance of *Rm-TWT* because the cost coefficients  $c'_{jkt}$  are non-negative and non-decreasing over time. However, the trivial augmentation is not necessarily feasible for every instance of *Rm-TWET*, and (2.23) might therefore be an invalid Benders cut. To illustrate, consider an instance of *Rm-TWET* and assume that for some assignment  $\bar{\mathbf{y}}$  of the jobs to the machines we solve  $(\mathbf{TR}_k - \mathbf{R})$  with  $H_k \leq d_j - 1$ , where  $j \notin J_k$  and  $p_{jk} > 1$ . In the trivially augmented solution for  $(\mathbf{DS}_k - \mathbf{F})$ , constraint (2.17) for job  $j$  and time period  $d_j$  is violated because  $c'_{jkd_j} = \frac{\epsilon_j}{p_{jk}} \left( \frac{1}{2} - \frac{p_{jk}}{2} \right) < 0$  for  $\epsilon_j > 0$  and  $p_{jk} > 1$ , and  $\bar{u}_{jk} + \bar{v}_{kd_j} = 0 + 0 \leq c'_{jkd_j}$  does not hold. Therefore, we need a mechanism which can always extend an optimal

solution of  $(\mathbf{DS}_k - \mathbf{R})$  to an optimal solution of  $(\mathbf{DS}_k - \mathbf{F})$ . Proposition 2.3 proves that the cut strengthening procedure described next fulfills this goal. This ensures that the dual slave problem is always solved over the same feasible region and the generated Benders cuts are valid.

Several papers in the literature report that a straightforward implementation of Benders decomposition yields a dismal performance from a computational point of view (Fischetti et al., 2010, Magnanti and Wong, 1981, Üster and Agrahari, 2011, Van Roy, 1986, Wentges, 1996). This is often rooted in the primal degeneracy in the cut generation subproblem which implies the existence of multiple optimal solutions to the dual slave problem. That is, possibly several alternate cuts may be generated based on the same master problem solution, and the particular choice has a profound impact on the computational performance. These concerns are also valid for us because the transportation problem suffers from a well-known primal degeneracy. To address these issues, we initially adapted the generic Benders cut strengthening method introduced recently by Fischetti et al. (2010) to our problem. These authors argue that identifying a small set of constraints in the subproblem that allows us to cut the current master solution is of practical interest to enhance the computational performance. To this end, they pose the cut generation subproblem as a pure feasibility problem and look for a *minimal infeasible subsystem* of small cardinality. However, applying this technique to our problem does not preserve the transportation problem structure in the cut generation subproblems. This results in substantially prolonged subproblem solution times with ultimately uncompetitive overall performance for Benders decomposition. Instead, here we follow an approach that is similar to those of Van Roy (1986) and Üster and Agrahari (2011) to strengthen our Benders cuts, which also resolves the issue pointed out in the previous paragraph regarding the validity of the cuts constructed based on an optimal solution of  $(\mathbf{DS}_k - \mathbf{R})$ . We reap great savings in solution time from this enhancement. In fact, our algorithm exhibits very poor convergence without this cut strengthening.

The key to showing the validity of our cut generation as well as strengthening the Benders cuts is to prove that we can always augment an optimal solution of  $(\mathbf{DS}_k - \mathbf{R})$  to obtain a feasible solution of  $(\mathbf{DS}_k - \mathbf{F})$  with the same objective function value. This would establish that the augmented solution is optimal for

$(\mathbf{DS}_k - \mathbf{F})$  because  $\bar{y}_{jk} = 0$  for all  $j \notin J_k$  and  $v_{kt} \leq 0$  for all  $t = H_k + 1, \dots, H$  (see Proposition 2.3). Compared to (2.21), the benefit is that we can produce a strengthened Benders cut of the form

$$\eta_k \geq \sum_{j=1}^n p_{jk} \bar{u}''_{jk} y_{jk} + \sum_{t=1}^H \bar{v}''_{kt} \quad (2.24)$$

from an optimal solution  $(\bar{u}''_k, \bar{v}''_k)$  of  $(\mathbf{DS}_k - \mathbf{F})$  so that  $\bar{u}''_{jk} \neq 0$  for  $j \notin J_k$  in general. We first need the following result to attain our goal.

**Lemma 2.2.** *There exists an optimal solution  $(\bar{u}'_k, \bar{v}'_k)$  to  $(\mathbf{DS}_k - \mathbf{R})$  such that*

$$\max_{t=1, \dots, H_k} \bar{v}'_{kt} = 0.$$

*Proof.* Assume that an optimal solution  $(\bar{u}_k, \bar{v}_k)$  to  $(\mathbf{DS}_k - \mathbf{R})$  is available. The claim holds trivially if there are idle periods in the schedule – which would typically be true for an instance of *Rm-TWET* – because for any idle period  $t$  we have  $\bar{v}_{kt} = 0$  due to complementary slackness. We set  $(\bar{u}'_k, \bar{v}'_k) = (\bar{u}_k, \bar{v}_k)$ .

Otherwise, assume that there is no idleness in the schedule, i.e.,  $H_k = \sum_{j \in J_k} p_{jk}$ . Define  $\bar{v}_k^{\max} = \max_{t=1, \dots, H_k} \bar{v}_{kt} \leq 0$  and construct a new solution  $\bar{u}'_{jk} = \bar{u}_{jk} - |\bar{v}_k^{\max}|$ ,  $j \in J_k$ ,  $\bar{v}'_{kt} = \bar{v}_{kt} + |\bar{v}_k^{\max}|$ ,  $t = 1, \dots, H_k$ . Observe that  $(\bar{u}'_k, \bar{v}'_k)$  belongs to the feasible region of  $(\mathbf{DS}_k - \mathbf{R})$  because

$$\bar{u}'_{jk} + \bar{v}'_{kt} = \bar{u}_{jk} - |\bar{v}_k^{\max}| + \bar{v}_{kt} + |\bar{v}_k^{\max}| = \bar{u}_{jk} + \bar{v}_{kt} \leq c'_{jkt}, \quad j \in J_k, \quad t = 1, \dots, H_k,$$

by the feasibility of  $(\bar{u}_k, \bar{v}_k)$  for  $(\mathbf{DS}_k - \mathbf{R})$ , and  $\bar{v}'_{kt} = \bar{v}_{kt} + |\bar{v}_k^{\max}| \leq 0$  for all  $t = 1, \dots, H_k$ , by the definition of  $\bar{v}_k^{\max}$ . Furthermore, the objective function value associated with  $(\bar{u}'_k, \bar{v}'_k)$  is identical to that of  $(\bar{u}_k, \bar{v}_k)$ :

$$\begin{aligned} \sum_{j \in J_k} p_{jk} \bar{y}_{jk} \bar{u}'_{jk} + \sum_{t=1}^{H_k} \bar{v}'_{kt} &= \sum_{j \in J_k} p_{jk} (\bar{u}_{jk} - |\bar{v}_k^{\max}|) + \sum_{t=1}^{H_k} (\bar{v}_{kt} + |\bar{v}_k^{\max}|) \\ &= \sum_{j \in J_k} p_{jk} \bar{u}_{jk} - |\bar{v}_k^{\max}| \sum_{j \in J_k} p_{jk} + \sum_{t=1}^{H_k} \bar{v}_{kt} + |\bar{v}_k^{\max}| H_k \\ &= \sum_{j \in J_k} p_{jk} \bar{y}_{jk} \bar{u}_{jk} + \sum_{t=1}^{H_k} \bar{v}_{kt}. \end{aligned}$$



Therefore,  $(\bar{\mathbf{u}}'_k, \bar{\mathbf{v}}'_k)$  is an alternate optimal solution, and

$$\max_{t=1, \dots, H_k} \bar{v}'_{kt} = \max_{t=1, \dots, H_k} \left\{ \bar{v}_{kt} + |\bar{v}_k^{\max}| \right\} = 0$$

by the definition of  $\bar{v}_k^{\max}$ . □

Assume that we are given an optimal solution  $(\bar{\mathbf{u}}'_k, \bar{\mathbf{v}}'_k)$  of  $(\mathbf{DS}_k - \mathbf{R})$  which satisfies the property in Lemma 2.2 and a corresponding Benders cut of the form (2.23). Clearly, we can always extend the planning horizon in  $(\mathbf{DS}_k - \mathbf{R})$  to  $1, \dots, H$ , and augment this optimal solution with zeros as necessary and still preserve the optimality. Therefore, without loss of generality assume that an augmented optimal solution  $(\bar{\mathbf{u}}'_k, \bar{\mathbf{v}}''_k)$  is available to  $(\mathbf{DS}_k - \mathbf{R})$ , where  $\bar{v}''_{kt} = \bar{v}'_{kt}$  for  $t = 1, \dots, H_k$ , and  $\bar{v}''_{kt} = 0$  for  $t = H_k + 1, \dots, H$ . Based on this augmented optimal solution, we next explain how an original Benders cut of the form (2.23) is strengthened, and then prove that this strengthened cut corresponds to an optimal solution of  $(\mathbf{DS}_k - \mathbf{F})$  and is therefore valid.

The variables  $u_{jk}$ ,  $j \notin J_k$ , do not appear in  $(\mathbf{DS}_k - \mathbf{R})$  and are implicitly assumed to be zero. Consequently, no term appears on the right hand side of a Benders cut (2.23) for the jobs that are assigned to other machines in the current relaxed master solution  $\bar{\mathbf{y}}$ . However,  $\bar{y}_{jk} = 0$  for all such jobs  $j \notin J_k$ , and we can produce a stronger cut by incorporating  $y_{jk}$ ,  $j \notin J_k$ , into the right hand side of (2.23) with positive coefficients  $p_{jk} \bar{u}''_{jk}$ ,  $j \notin J_k$ , if possible. In order to compute a good set of values  $\bar{u}''_{jk}$ ,  $j \notin J_k$ , we solve the following optimization problem for a given augmented optimal solution  $(\bar{\mathbf{u}}'_k, \bar{\mathbf{v}}''_k)$  of  $(\mathbf{DS}_k - \mathbf{R})$ :

$$\text{maximize} \quad \sum_{j \notin J_k} p_{jk} u_{jk} \tag{2.25}$$

$$\text{subject to} \quad u_{jk} \leq c'_{jkt} - \bar{v}''_{kt}, \quad j \notin J_k, t = 1, \dots, H. \tag{2.26}$$

The constraints (2.26) are required to establish that the coefficients of the strengthened cut correspond to an optimal solution of  $(\mathbf{DS}_k - \mathbf{F})$  – see Proposition 2.3. Clearly, (2.25)-(2.26) decomposes by job, and the optimal solution is determined

as:

$$\bar{u}''_{jk} = \min \left\{ \min_{t=1, \dots, H_k} (c'_{jkt} - \bar{v}''_{kt}), \min_{t=H_k+1, \dots, H} c'_{jkt} \right\}, \quad j \notin J_k. \quad (2.27)$$

For an instance of *Rm-TWT*, the cost coefficients  $c'_{jkt}$  are non-decreasing over  $t = 1, \dots, H$ . In addition, we have  $\max_{t=1, \dots, H_k} \bar{v}''_{kt} = 0$ . Then,

$$\min_{t=1, \dots, H_k} (c'_{jkt} - \bar{v}''_{kt}) \leq \max_{t=1, \dots, H_k} c'_{jkt} \leq \min_{t=H_k+1, \dots, H} c'_{jkt}. \quad (2.28)$$

Consequently, (2.27) simplifies to

$$\bar{u}''_{jk} = \min_{t=1, \dots, H_k} (c'_{jkt} - \bar{v}''_{kt}), \quad j \notin J_k, \quad (2.29)$$

for *Rm-TWT*.

For *Rm-TWET*, we have to differentiate between two cases because the cost coefficients  $c'_{jkt}$ ,  $1, \dots, H$ , are not non-decreasing over time:

$$\bar{u}''_{jk} = \left\{ \begin{array}{ll} \min \left( \min_{t=1, \dots, H_k} (c'_{jkt} - \bar{v}''_{kt}), c'_{jkH_{k+1}} \right) & \text{if } H_k \geq d_j \\ \min \left( \min_{t=1, \dots, H_k} (c'_{jkt} - \bar{v}''_{kt}), c'_{jkd_j} \right) & \text{if } H_k \leq d_j - 1 \end{array} \right\}, \quad j \notin J_k. \quad (2.30)$$

Thus, the strengthened cut finally takes the form specified in (2.24), where  $\bar{u}''_{jk} = \bar{u}'_{jk}$  for  $j \in J_k$  and  $\bar{u}''_{jk}$ ,  $j \notin J_k$ , is calculated based on either (2.29) or (2.30), respectively, depending on whether we solve an instance of *Rm-TWT* or *Rm-TWET*. We next prove that this augmented solution  $(\bar{u}''_k, \bar{v}''_k)$  is optimal for  $(\mathbf{DS}_k - \mathbf{F})$ .

**Proposition 2.3.** *The dual variables  $(\bar{u}''_k, \bar{v}''_k)$ , which produce a strengthened Benders cut (2.24), are optimal with respect to  $(\mathbf{DS}_k - \mathbf{F})$ .*

*Proof.* Recall that  $(\bar{u}''_k, \bar{v}''_k)$  is constructed by augmenting an optimal solution  $(\bar{u}'_k, \bar{v}'_k)$  of  $(\mathbf{DS}_k - \mathbf{R})$  which satisfies the property in Lemma 2.2. Therefore,  $\bar{u}''_{jk} + \bar{v}''_{kt} \leq c'_{jkt}$ ,  $j \in J_k, t = 1, \dots, H_k$ , and  $\bar{v}''_{kt} \leq 0$ ,  $t = 1, \dots, H_k$ , hold automatically. In addition,  $\bar{v}''_{kt}$ ,  $t = H_k + 1, \dots, H$ , are set directly to zero. Therefore, we only need to verify that  $\bar{u}''_{jk} + \bar{v}''_{kt} \leq c'_{jkt}$ ,  $j \in J_k, t = H_k + 1, \dots, H$ , and  $\bar{u}''_{jk} + \bar{v}''_{kt} \leq c'_{jkt}$ ,  $j \notin J_k, t = 1, \dots, H$ , to show the feasibility of  $(\bar{u}''_k, \bar{v}''_k)$  for  $(\mathbf{DS}_k - \mathbf{F})$ . The latter inequalities are enforced directly by the constraints (2.26). For the former, note that for any job  $j \in J_k$  the

end of the planning horizon  $H_k$  is larger than  $d_j$  in both *Rm-TWT* and *Rm-TWET*. Then, by a similar argument that leads to (2.28),  $\bar{u}''_{jk} \leq \max_{t'=1, \dots, H_k} c'_{jkt'}$  and we obtain  $\bar{u}''_{jk} + \bar{v}''_{kt} = \bar{u}''_{jk} \leq \max_{t'=1, \dots, H_k} c'_{jkt'} \leq c'_{jkt}$  for all time periods  $t = H_k + 1, \dots, H$ , as desired.

The optimal objective function value of  $(\mathbf{DS}_k - \mathbf{F})$  is bounded from above by that of  $(\mathbf{DS}_k - \mathbf{R})$  because all constraints of  $(\mathbf{DS}_k - \mathbf{R})$  are present in (2.17)-(2.18),  $\bar{y}_{jk} = 0$  for  $j \notin J_k$ , and  $\sum_{t=H_k+1}^H v_{kt} \leq 0$ . This completes the proof since the objective function value associated with  $(\bar{u}''_k, \bar{v}''_k)$  in  $(\mathbf{DS}_k - \mathbf{F})$  is clearly identical to that associated with the optimal solution  $(\bar{u}'_k, \bar{v}'_k)$  in  $(\mathbf{DS}_k - \mathbf{R})$ .  $\square$

The pseudo-code of our Benders decomposition scheme with the cut strengthening feature for solving  $(\mathbf{TR} - \mathbf{A})$  is stated in Algorithm 2 where Procedure *generate\_cuts* is stated in Algorithm 1, and Proposition 2.3 proves its correctness. The cut strengthening specified by the Steps 3–6 in Algorithm 1 has a pseudo-polynomial time complexity of  $O(nH)$  with an overall complexity of  $O(mnH)$  for  $m$  machines. In practice, it is very fast.

In classical textbook applications of Benders decomposition, the current relaxed master problem is solved to optimality and then cuts generated based on this optimal solution are added to it before the relaxed master problem is re-optimized. This loop is repeated until the optimality gap of  $(\mathbf{RMP})$  – the expression  $\frac{z(\bar{\mathbf{y}}) - \sum_{k=1}^m \bar{\eta}_k}{\sum_{k=1}^m \bar{\eta}_k}$  – is smaller than a prespecified tolerance level, where the current optimal objective  $\sum_{k=1}^m \bar{\eta}_k$  of  $(\mathbf{RMP})$  is a lower bound on that of  $(\mathbf{TR} - \mathbf{A})$  and  $z(\bar{\mathbf{y}})$  is the objective value of a feasible solution of  $(\mathbf{TR} - \mathbf{A})$ . The primary drawback of this classical scheme is that a new search tree is constructed every time the relaxed master problem is solved (Rubin, 2011). Consequently, valuable time may be expended toward re-evaluating the same nodes over and over again. In contrast, using the *lazy constraint* technology offered by the state-of-the-art solvers allows us to execute the entire algorithm on a single search tree (IBM ILOG CPLEX, 2011). In Step 11 of Algorithm 1, we invoke the *lazy constraint callback* routine for every candidate incumbent solution. The callback routine either identifies a missing Benders cut violated by the candidate solution and introduces it as a lazy constraint into the model or certifies the candidate as valid. Ultimately, no integer solution is evaluated multiple times during the course of the algorithm. Moreover,

---

**Algorithm 2:** Solving  $(\text{TR} - \mathbf{A})$  by Benders decomposition and lazy constraint generation.

---

```

// Initialization
1 Create (RMP) with (2.19), (2.20), (2.22). Add the load balancing constraints (2.14)
  for  $Rm$ -TWT;
2 repeat // To improve the initial objective value of (RMP).
3   Construct a feasible assignment  $\bar{\mathbf{y}}$  of jobs to  $m$  machines by some heuristic.
4    $[\text{cuts}, z_1(\bar{\mathbf{y}}), \dots, z_m(\bar{\mathbf{y}})] = \text{generate\_cuts}(\bar{\mathbf{y}})$ ; //  $\text{cuts}$  is a collection of  $m$ 
   cuts.
5   Add  $\text{cuts}$  to (RMP) as lazy constraints;
6 until some termination condition is satisfied; // We run a simple dispatch rule
   once.

// Main loop
7 Invoke CPLEX on (RMP);
8 repeat
9   Identify a new incumbent candidate  $\bar{\mathbf{y}}$  with an objective value of  $\sum_{k=1}^m \bar{\eta}_k$ ;
10   $\text{accept\_candidate} = \text{true}$ ;
11   $[\text{cuts}, z_1(\bar{\mathbf{y}}), \dots, z_m(\bar{\mathbf{y}})] = \text{generate\_cuts}(\bar{\mathbf{y}})$ ; //  $\text{cuts}$  is a collection of  $m$ 
   cuts.
12  for  $k = 1$  to  $m$  do
13    if  $\bar{\eta}_k < z_k(\bar{\mathbf{y}})$  then //  $\bar{\mathbf{y}}$  violates some of the missing Benders cuts.
14    |   Add  $\text{cuts}_k$  to (RMP) as a lazy constraint,  $\text{accept\_candidate} = \text{false}$ ;
15 until CPLEX determines that the relative optimality gap of the current incumbent is less
   than some threshold;
16 The best available job partition  $\bar{\mathbf{y}}^*$  for  $(\text{TR} - \mathbf{A})$  is retrieved from CPLEX. If desired,
   the associated preemptive machine schedules are obtained by solving  $(\text{TR}_k - \mathbf{R})$ 
   with  $\bar{\mathbf{y}}^*$  for  $k = 1, \dots, m$ .

```

---

labeling the generated cuts as lazy informs the solver that most of such constraints are not expected to be active at the optimal solution. Thus, we fully exploit the capabilities of the solver and allow it to apply the generated cuts as it deems necessary. The use of the lazy constraint technology appears to be relatively rare in the operations research literature, and we hope that it will be employed more frequently in the future given that it may unleash the power of a cut generation algorithm which seems impractical otherwise.

## 2.5 Computational Results

Outstanding among the accomplishments of this research is that both  $Rm$ -TWT with a regular scheduling objective (see Section 2.5.1) and  $Rm$ -TWET with a non-regular scheduling objective (see Section 2.5.2) are tackled successfully by the

exact same solution approach. For both problems, the overarching goal of our computational study is to demonstrate that the proposed Benders-type method solves the preemptive relaxation (**TR – A**) to (near-)optimality in short computation times and provides tight lower bounds as well as high quality job partitions for the original problems. Very large instances of both problems are within the reach of our algorithm; however, we concede that the performance is somewhat better for *Rm-TWT* than for *Rm-TWET*.

The size of an instance is determined by the parameters  $m$  and  $n'$  so that the number of jobs is set to  $n = mn'$ . For each job  $j \in \{1, \dots, n\}$ , the processing time  $p_{j1}$  on the first machine is randomly drawn from the discrete uniform distribution  $U[p_{\min}, p_{\max}]$ . The processing times  $p_{jk}$  for  $k \in \{2, \dots, m\}$  are then created as  $\max(1, \lfloor U[1 - \theta, 1 + \theta] p_{j1} \rfloor)$ . The earliness weight per unit time  $\epsilon_j$  is generated from a discrete uniform distribution  $U[\epsilon_{\min}, \epsilon_{\max}]$ , and the corresponding unit tardiness weight is computed as  $\lfloor U[\alpha, \beta] \epsilon_j \rfloor$ . For *Rm-TWT*, all unit earliness weights are then set to zero. We generate the due dates by following the method of [Potts and van Wassenhove \(1982\)](#), which is a popular scheme in the literature ([Liaw et al., 2003](#), [Lin et al., 2011](#), [Shim and Kim, 2007a](#)). The integral due date  $d_j$  of job  $j$  is calculated as  $\lfloor U[\bar{P}(1 - \text{TF} - \frac{\text{RDD}}{2})^+, \bar{P}(1 - \text{TF} + \frac{\text{RDD}}{2})] \rfloor$ , where the tardiness factor TF controls the tightness of the due dates and the due date range factor RDD determines their spread.  $\bar{P} = \sum_j \sum_k p_{jk} / m^2$  may be considered as the average load per machine. The parameters of the instance generation procedure are summarized in [Table 2.2](#).

**Table 2.2** Instance generation parameters.

$m$	$n'$	$[p_{\min}, p_{\max}]$	$\theta$	$[\epsilon_{\min}, \epsilon_{\max}]$	$[\alpha, \beta]$	TF	RDD
{2, 3, 4, 5}	{20, 30, 40}	[25, 100]	0.25	[1, 10]	[1.5, 3.0]	{0.4, 0.6, 0.8, 1.0}	{0.2, 0.4, 0.6}

There are 12 combinations of the TF, RDD values and for each combination, 5 instances are generated. Therefore, we create 60 instances for each pair of  $m$ ,  $n'$  values and a total of 720 instance pairs. The instances in a pair are identical, except that  $\epsilon_j = 0, j = 1, \dots, n$ , in the *Rm-TWT* instance. This data generation scheme allows us to draw clear conclusions about the relative difficulty of *Rm-TWET* with respect to *Rm-TWT*. As pointed out by [Kedad-Sidhoum et al. \(2008\)](#), the motivation for the relatively large TF and small RDD values is that in most

practical production environments the due dates are not loose and not distant from each other. The rationale behind the selected  $[\alpha, \beta]$  values reflects that the earliness cost is typically regarded as a finished goods inventory holding cost and should be less than the cost of loss of customer goodwill or a contractual penalty represented by the tardiness cost.

The computational results are obtained on a personal computer with a 3.80 GHz Intel Core i7 920 CPU with Hyper-Threading enabled and 24 GB of memory running on Windows 7. Algorithms 1-2, which are collectively referred to as **(TR – A)-BDS** in this section, were implemented in C++ using the Concert Technology component library of IBM ILOG CPLEX 12.4. The cut generation procedure in Algorithm 2 is parallelized through the Boost 1.51 library. More specifically, when a new integer feasible solution is identified in the search tree for **(RMP)**,  $m$  threads are constructed in the lazy constraint callback routine to solve **(TR<sub>k</sub>)**,  $k = 1, \dots, m$ , in parallel. Note that in the presence of a control callback – such as the lazy constraint callback in **(TR – A)-BDS** – CPLEX applies a traditional branch-and-cut strategy by switching off its dynamic search feature and operates in an opportunistic parallel search mode. Following the termination of **(TR – A)-BDS**, we call the **SiPS/SiPSi**<sup>1</sup> libraries (Tanaka and Fujikuma, 2012, Tanaka et al., 2009) to solve  $m$  single-machine problems for each job partition present in the final “solution pool” of CPLEX and obtain feasible solutions for  $Rm$ -TWT and  $Rm$ -TWET. Note that the current CPLEX engine generates and keeps multiple feasible solutions in addition to the optimal solution in a solution pool to help the user choose one that may fit criteria not represented explicitly in the current model solved (IBM ILOG CPLEX, 2011). Furthermore, to promote the quality of the job partitions, the switch **MIPEmphasis** in **(TR – A)-BDS** is set to 4 in order to urge CPLEX “to apply considerable additional effort toward finding high quality feasible solutions that are difficult to locate” (IBM ILOG CPLEX, 2011). The source code of our algorithms and the test instances are available [at this location](#)<sup>2</sup>.

To justify the use of the proposed Benders-type approach to solve **(TR – A)**, we benchmark it against **(TR – A)-CPX**, where the monolithic formulation **(TR – A)** is solved directly by invoking CPLEX. In this case, we let CPLEX decide whether

<sup>1</sup><https://sites.google.com/site/shunjitanaka/sips/>

<sup>2</sup><http://people.sabanciuniv.edu/bulbul/publications.html>

to apply its dynamic search by running it with the default parameter settings, except that the opportunistic parallel search mode is turned on for a head-to-head comparison with **(TR – A)-BDS**. The relative gap tolerance parameter `EpGap` of CPLEX is set to 3% while solving **(TR – A)** by either **(TR – A)-CPX** or **(TR – A)-BDS**. In addition, to illustrate the value of our approach in the absence of scalable alternate solution approaches for *Rm-TWT* and *Rm-TWET* in the literature and be able to provide more accurate optimality gaps for our lower and upper bounds, we also solve a time-indexed integer programming formulation for *Rm-TWT* and *Rm-TWET* via CPLEX under the default parameter settings. This formulation is referred to as **(TI)** in the sequel and obtained from that in [Kedad-Sidhoum et al. \(2008\)](#) in a straightforward way by augmenting the time-indexed variables with a machine index and imposing a machine capacity constraint for each combination of machine and time period so that no more than one job is in process at any time instant on any machine. The best lower bound retrieved from **(TI)** at termination provides an alternate lower bound for the original non-preemptive problems, and the best available objective value at termination provides us with a benchmark for the non-preemptive solutions we construct for *Rm-TWT* and *Rm-TWET*.

All formulations are solved within the same working memory limit of 15 GB (`WorkMem = 15000`). However, the memory footprint of **(TR – A)-BDS** does not exceed a few gigabytes even for the largest instances with 200 jobs and 5 machines. The maximum number of threads that CPLEX is allowed to use – governed by the parameter `Threads` – is seven for all methods. The time limit parameter `TiLim` takes on the values 1800, 1800 and 600 seconds for **(TI)**, **(TR – A)-CPX**, and **(TR – A)-BDS**, respectively.

The next section reports the results obtained for *Rm-TWT*, and the results for *Rm-TWET* are relegated to Section 2.5.2. For ease of perusal, all tables employ a color formatting scheme so that the values of a performance indicator ranging from better to worse are indicated with colors changing from green towards red.

### 2.5.1 Results for *Rm-TWT*

Table 2.3 consists of 12 parts, one for each possible combination of  $n$  and  $m$  listed in the first two columns. We report three types of percentage gaps in the table,

labeled as “(TR – A)-BDS”, “LB Quality”, and “Feasible Sol’n” in Columns 4–12. The average times needed to solve the preemptive relaxation (TR – A) to within 3% of optimality by (TR – A)-CPX and (TR – A)-BDS are presented in Columns 13–18. The color formatting is applied to these two sets of columns together to facilitate a head-to-head comparison. For each performance indicator, detailed results for each possible combination of TF and RDD values are included. The TF values appear in the third column, and the RDD values are specified in the column headers. All gaps larger than 100% are set to 100%, and the gap of a feasible solution with a positive objective function value with respect to a lower bound of zero is assumed to be 100%. Each value in the table represents an average over five instances based on our data generation scheme discussed previously.

**Table 2.3** Results for *Rm-TWT*.

<i>n</i>	<i>m</i>	TF	RDD		Percentage Gaps						(TR – A) - Solution Times						
			(TR – A)-BDS			LB Quality			Feasible Sol’n			CPX			BDS		
			0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
40	2	0.4	1.3	1.6	0.5	3.0	7.5	13.2	1.6	4.5	6.3	2	3	10	2	2	3
		0.6	2.0	2.2	2.2	2.8	3.5	6.4	1.0	1.8	2.2	4	3	4	2	2	3
		0.8	2.5	2.6	2.4	2.8	3.6	3.7	0.8	0.6	0.6	6	5	5	1	1	3
		1.0	2.4	2.1	2.2	2.3	2.4	2.3	0.5	0.5	0.8	6	6	5	1	1	1
60	2	0.4	2.2	1.8	1.6	3.5	5.3	9.5	0.9	2.0	5.0	8	7	11	5	7	10
		0.6	2.4	2.4	2.6	3.3	4.4	5.7	0.9	1.3	1.7	15	12	11	5	7	18
		0.8	2.7	1.7	2.5	3.1	2.7	4.4	1.2	0.6	1.2	21	19	17	3	6	9
		1.0	2.2	2.8	2.8	1.9	2.2	2.5	0.6	1.2	1.0	24	25	24	2	2	2
80	3	0.4	2.4	2.1	1.4	5.0	9.3	42.9	1.9	3.6	36.8	20	60	215	2	4	4
		0.6	2.7	2.7	2.8	3.8	5.0	8.7	1.5	1.7	3.0	32	30	36	1	2	12
		0.8	2.8	2.8	2.6	3.1	4.3	4.8	1.4	0.9	0.9	40	40	36	1	2	3
		1.0	2.6	2.5	2.5	2.2	2.6	3.2	0.8	0.6	0.8	41	44	41	1	1	1
90	3	0.4	2.0	2.2	2.0	3.0	5.0	11.6	1.1	2.0	6.8	15	12	44	8	17	20
		0.6	2.2	1.4	2.6	2.9	2.8	4.2	1.0	1.1	0.9	33	27	30	10	21	36
		0.8	2.9	2.3	2.7	3.3	3.0	4.3	1.5	0.7	1.1	52	56	44	3	9	10
		1.0	2.2	2.4	2.3	1.6	2.7	2.7	1.1	0.4	0.6	69	63	59	3	6	7
90	4	0.4	2.7	2.6	1.2	6.8	10.4	38.0	3.7	5.0	36.6	42	295	1716	5	7	11
		0.6	2.7	2.5	3.8	4.2	5.5	10.1	1.6	2.1	4.2	79	73	484	2	7	462
		0.8	2.8	2.4	2.9	3.6	4.3	5.9	1.1	0.7	1.0	111	104	98	1	6	11
		1.0	2.5	2.8	2.8	2.2	3.1	3.7	0.7	1.1	0.7	118	112	121	1	1	2
90	3	0.4	2.2	2.2	0.9	3.8	7.3	26.9	1.6	3.5	21.2	34	112	1030	5	11	28
		0.6	2.6	2.8	2.9	3.3	4.3	6.3	1.2	1.4	2.1	93	87	93	2	5	24
		0.8	2.6	2.5	2.6	3.3	3.8	4.5	1.1	0.9	1.2	122	121	117	2	3	5
		1.0	2.6	2.4	2.4	2.1	3.0	3.2	1.2	0.8	1.0	148	142	138	2	2	2

Continued on next page...



Table 2.3 continued...

$n$	$m$	RDD TF	Percentage Gaps									(TR – A) - Solution Times					
			(TR – A)-BDS			LB Quality			Feasible Sol'n			CPX			BDS		
			0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
100	5	0.4	2.7	2.7	2.8	7.3	13.7	20.0	6.2	13.3	20.0	104	1119	1800	6	13	125
		0.6	2.7	2.8	5.9	4.2	5.7	12.8	2.9	4.6	9.3	149	168	883	4	15	601
		0.8	2.6	2.8	3.1	3.7	4.9	6.5	2.3	3.5	4.9	235	182	242	2	15	308
		1.0	2.5	2.5	2.8	2.8	3.3	4.3	1.4	1.9	2.7	253	233	235	2	2	3
120	3	0.4	2.5	1.9	1.1	3.7	6.5	42.5	2.3	5.8	42.5	48	66	1367	7	20	109
		0.6	2.2	2.6	2.4	2.9	4.2	5.6	1.6	2.9	4.6	205	172	162	4	8	49
		0.8	2.6	2.9	2.7	3.0	4.0	4.5	1.5	2.3	2.8	262	254	221	4	5	8
		1.0	2.8	2.3	2.6	3.0	2.7	3.3	1.1	1.3	1.4	338	344	303	3	4	6
120	4	0.4	2.2	2.9	10.8	4.1	9.1	20.0	3.0	7.9	20.0	96	516	1816	13	18	135
		0.6	2.7	2.6	3.1	3.6	4.4	7.4	2.4	2.8	5.9	205	209	327	4	18	225
		0.8	2.5	2.5	2.9	3.2	4.1	5.0	1.9	2.6	3.2	316	315	287	3	6	16
		1.0	2.7	2.7	2.7	3.0	3.2	3.7	1.4	1.6	2.3	346	326	291	2	3	3
150	5	0.4	2.5	2.9	0.0	5.2	10.0	0.0	4.0	9.0	0.0	216	1356	1527	19	35	18
		0.6	2.7	2.7	3.8	3.7	4.6	8.8	2.1	3.3	6.6	380	379	804	5	39	415
		0.8	2.6	2.7	2.9	3.3	4.3	5.5	1.8	2.8	4.1	683	646	600	4	11	48
		1.0	1.3	2.5	2.4	1.6	3.1	3.5	0.8	1.7	2.3	752	713	653	5	4	4
160	4	0.4	2.8	2.8	0.0	4.5	9.2	0.0	3.1	7.9	0.0	143	451	1361	9	40	29
		0.6	2.7	2.9	2.9	3.3	4.2	6.4	1.6	2.7	4.8	429	302	704	6	11	176
		0.8	2.6	2.2	2.6	3.1	3.3	4.3	1.6	2.2	2.8	713	742	668	5	9	12
		1.0	2.0	2.0	2.6	2.2	2.4	3.3	0.9	1.1	1.8	934	836	851	5	6	6
200	5	0.4	2.5	3.4	0.0	4.5	12.3	0.0	3.3	10.8	0.0	345	1476	793	24	324	35
		0.6	2.5	2.8	3.9	3.2	4.5	7.9	1.7	2.7	5.4	964	752	1327	11	33	510
		0.8	2.6	2.7	2.7	3.1	3.9	4.6	1.9	2.2	3.0	1720	1708	1440	7	29	36
		1.0	2.4	2.0	2.3	2.7	2.4	3.1	2.5	1.8	2.1	1803	1771	1719	7	9	9

The optimality gaps depicted in Columns 4–6 are retrieved from CPLEX at the termination of (TR – A)-BDS, where CPLEX computes the optimality gap by taking the ratio of the best available lower bound to the objective value associated with the best integer solution at termination and then subtracting this ratio from 1. These results indicate that (TR – A)-BDS is able to solve the preemptive relaxation to the targeted precision of 3%. More specifically, (TR – A)-BDS terminates due to the time limit of 600 seconds for only 22 instances out of a total of 720. The corresponding number for (TR – A)-CPX is 47 with a time limit of 1800 seconds. The average (& median) gaps of (TR – A)-BDS and (TR – A)-CPX for those instances that could not be solved within the specified time limits are 7.22% (& 4.05%) and 74.14% (& 100%), respectively. Therefore, we conclude that the use of our Benders-type method for solving (TR – A) is well-justified.

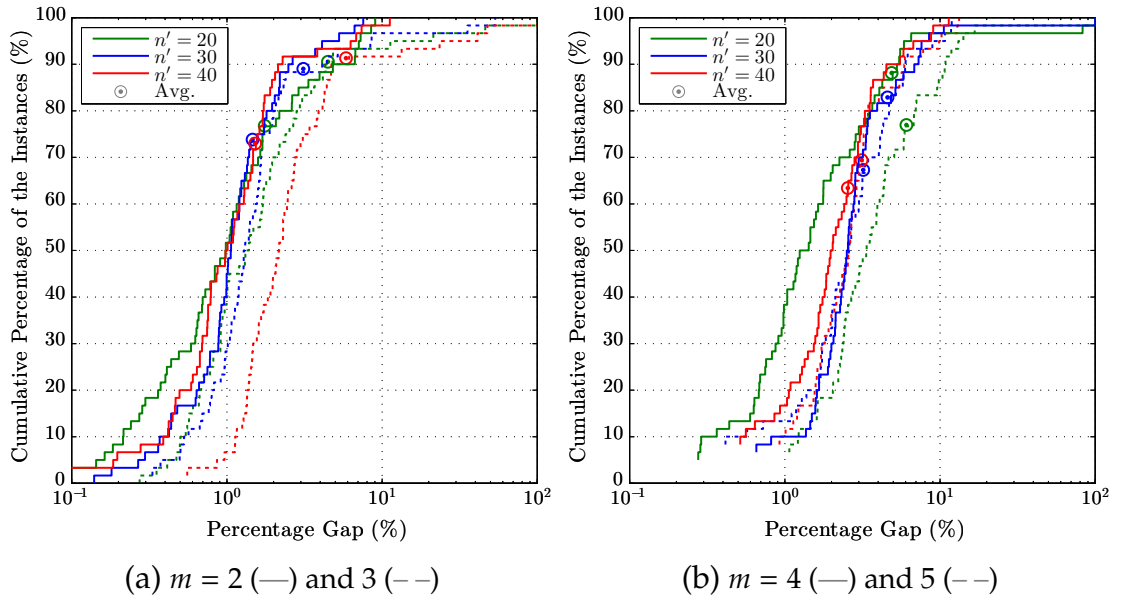
The next three columns under “LB Quality” attest to the quality of the lower bound (LB) provided by (TR – A)-BDS for the optimal objective value of  $Rm-TWT$ .

For a given instance, the expression  $\frac{(\text{“Best Integer”} - \text{“LB”})}{(\text{“Best Bound”})}$  provides an upper bound on the gap of LB, where “Best Integer” and “Best Bound” are the objective function values associated with the best feasible solution available – retrieved from either our approach or (TI) – and the best lower bound provided by any one of the methods (TR – A)-CPX, (TR – A)-BDS, or (TI), respectively. For any  $n, m$  combination, the average LB gap summarized across all TF and RDD values does not exceed 8.15%, and the average LB gap across all instances is just 5.64%. In fact, only 8% of the instances (58 instances) have an LB gap larger than 10%.

The following three columns under “Feasible Sol’n” present the average upper bounds on the optimality gaps attained by our non-preemptive feasible solutions. For a given feasible solution, an upper bound on the optimality gap is calculated as  $\frac{(\text{“OFV”} - \text{“Best Bound”})}{(\text{“Best Bound”})}$ , where “OFV” is the objective function value of the feasible solution. We do not include detailed results about (TI) but note that the incumbent from (TI) is hardly competitive with the best feasible solution obtained from (TR – A)-BDS, except for the 40-job instances. Moreover, even the LP relaxation of (TI) is not solved within half an hour for instances with 100 or more jobs. The average (& median) optimality gaps over all instances solved are 3.55% (& 1.73%) and 30.28% (& 10.20%) for (TR – A)-BDS and (TI), respectively. Perhaps more importantly, the proposed approach delivers a robust performance and scales to very large instances. With the exception of a little over 4% of the instances (31 out of 720), the optimality gap is always below 10%. The corresponding number for (TI) is 50% (181 out of 360).

The relatively higher gaps under “LB Quality” and “Feasible Sol’n” in Table 2.3 for TF = 0.4 stem from the small objective function values associated with loose due dates. Even small errors result in large percentage gaps in this case. Note that the objective function value of an instance with TF = 0.6, 0.8, and 1.0 is on average 7.5, 25.1, and 45.9 times larger, respectively, compared to that of an instance with TF = 0.4. A second contributing factor here is the growing size of (TI) with looser due dates. Frequently, even the LP relaxation is not solved within the allotted time for such instances, and this results in smaller “Best Bound” values in general. In other words, the actual performance for TF = 0.4 is probably better than what it appears to be.

The robustness of the quality of the feasible solutions obtained from our



**Figure 2.1** The empirical distributions of the optimality gaps of the upper bounds by **(TR – A)-BDS** for  $Rm$ -TWT.

Benders-type approach is further illustrated in Figures 2.1a–2.1b. The empirical distributions of the optimality gaps of the feasible solutions associated with **(TR – A)-BDS** are plotted in these figures. The horizontal axes are in logarithmic scale to increase the readability of the graph. Note that the median percentage gap for each curve corresponds to the 50% mark on the vertical axis, and the average gaps are explicitly indicated. The curves are clustered and rise steeply. That is, the quality of the partitions retrieved from **(TR – A)-BDS** is not particularly sensitive to the increasing number of jobs  $n'$  per machine.

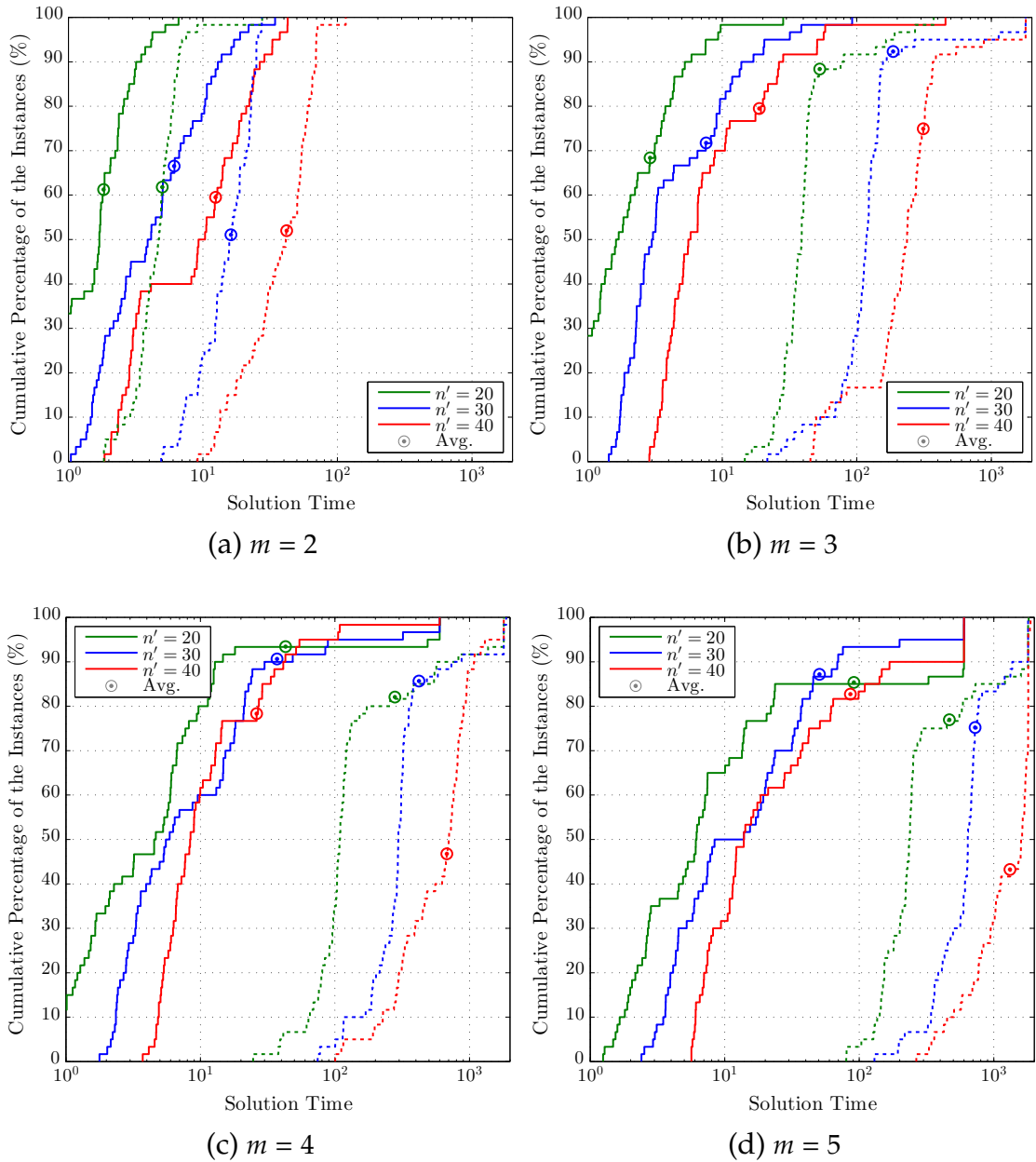
The solution time performance of **(TR – A)-BDS** is overwhelmingly superior to that of **(TR – A)-CPX**. Based on the instances that are solved by both methods within the time limit, the ratio of the solution time of **(TR – A)-CPX** to that of **(TR – A)-BDS** is 46.7 on average. Out of a total of 720 instances, only 35 of them take slightly more time to solve for **(TR – A)-BDS** compared to **(TR – A)-CPX**. For both methods, instances with loose average due dates within a relatively wide range are more problematic. However, tightening the due dates does also hurt the performance of **(TR – A)-CPX** while it benefits that of **(TR – A)-BDS**.

The empirical distributions of the solution times of **(TR – A)-BDS** and **(TR – A)-CPX** are plotted with solid and dashed lines in Figure 2.2, respectively. Similar to those in Figure 2.1, the horizontal axes are in logarithmic scale. The performance of **(TR – A)-CPX** is adversely affected by both an increasing number of

machines  $m$  and an increasing number of jobs per machine  $n'$  in an instance. To make the former observation concrete, note that the percentage of the instances with  $n' = 20$  solved to optimality by **(TR – A)-CPX** within 60 seconds is 100%, 88.3%, 6.7%, and 0% for  $m = 2, 3, 4, 5$ , respectively. In comparison, **(TR – A)-BDS** obtains the optimal solution for 100%, 98.9%, 93.3%, and 82.8% of the instances with  $m = 2, 3, 4, 5$ , respectively, in less than 60 seconds. Note that these latter numbers are aggregated over  $n'$ , including larger instances with  $n' = 30, 40$  as well. Clearly, **(TR – A)-BDS** displays a significantly more stable performance. Finally, we note that the solution times of **(TR – A)-BDS** are strongly correlated with the number of Benders cuts generated, as expected. The median percentage of the active Benders cuts for the final node problem in the search tree is 86.4% with a corresponding average of 81.3%.

Recall that we call the **SiPS/SiPSi** libraries ([Tanaka and Fujikuma, 2012](#), [Tanaka et al., 2009](#)) to solve  $m$  single-machine problems for each job partition present in the final solution pool of CPLEX and obtain feasible solutions for  $Rm$ -TWT and  $Rm$ -TWET following the termination of **(TR – A)-BDS**. We do not report detailed results for the sake of brevity, but our use of an optimal algorithm to solve the single-machine problems for a given job partition is well-justified. Even for the five machine and 200 job instances, it takes an average of 2.31 seconds and no more than 6.96 seconds to solve all single-machine problems to optimality by the **SiPS/SiPSi** solver for all job partitions identified. This solver is extremely fast; the time expended for a 40-job single-machine instance is about 30 milliseconds. We emphasize that the best solution of the preemptive relaxation does not necessarily produce the best non-preemptive solution for the original problem. Therefore, the ability of locating many high-quality job partitions in the search tree is a critical advantage of **(TR – A)-BDS**, which identifies on average 4.7 times more job partitions per instance compared to **(TR – A)-CPX**. This characteristic may also prove useful in order to jump start a population based heuristic following the completion of **(TR – A)-BDS**. In summary, coupled with its demonstrated ability to construct high-quality lower and upper bounds for the original problem, the outstanding total solution time performance of our approach makes it a viable alternative for tackling very large instances of  $Rm$ -TWT successfully.

We conclude this section with a brief discussion on the time-indexed formula-



**Figure 2.2** The empirical distributions of the solution times of  $(\text{TR} - \text{A})\text{-BDS}$  and  $(\text{TR} - \text{A})\text{-CPX}$  for  $Rm\text{-TWT}$ .

tion **(TI)**. The main purpose of solving **(TI)** in this work is to incorporate the objective function value of the incumbent solution and the best lower bound available at termination into the “Best Integer” and “Best Bound” values, respectively, so that we quantify the gaps of our lower and upper bounds as accurately as possible. Otherwise, solving **(TI)** is not a scalable solution approach for  $Rm\text{-TWT}$  as we discussed previously in this section. In addition, the LP relaxation of **(TI)** does also suffer from the same scalability issue as a lower bounding method. We provide further specifics and settle this issue below.

On the one hand, the LP relaxations of time-indexed formulations are strong and provide very tight bounds. On the other hand, however, the size of a time-indexed formulation grows with the length of the planning horizon and is therefore pseudo-polynomial. From a computational perspective, the solution effort expended increases rapidly with longer processing times, and CPLEX cannot solve the LP relaxation of **(TI)** or find any feasible solution within the time limit of 1800 seconds for the *Rm-TWT* (and *Rm-TWET*) instances with greater than 90 jobs. For the sake of completeness, we benchmarked the lower bound produced by **(TR – A)-BDS** against the optimal objective function value of the LP relaxation of **(TI)**. These two lower bounds do not dominate each other. There are instances in which the objective value of the LP relaxation of **(TI)** is larger than that of **(TR – A)** and vice versa. The best lower bound retrieved from **(TR – A)-BDS** at termination is on average 97.04% of the optimal objective value of the LP relaxation of **(TI)**, computed over 360 *Rm-TWT* instances with  $n \leq 90$ . The corresponding figure for the *Rm-TWET* instances is 94.29%. Furthermore, recall that **(TR – A)-BDS** terminates with a 3% relative optimality gap. Therefore, it is fair to state that the lower bounds provided by **(TR – A)-BDS** and the LP relaxation of **(TI)** are of comparable quality. Ultimately, **(TR – A)-BDS** is the clear choice as a lower bounding technique given its superior computational time performance and the high quality of the non-preemptive schedules based on the solution of **(TR – A)-BDS**.

### 2.5.2 Results for *Rm-TWET*

Table 2.4 is structured identically to Table 2.3 in Section 2.5.1 and depicts the percentage gap and solution time results for *Rm-TWET*. Unsurprisingly, both solving the preemptive relaxation and obtaining high-quality non-preemptive solutions pose a more difficult challenge for *Rm-TWET* than for *Rm-TWT*. In general, the gaps are larger and the solution times are longer than those in Section 2.5.1. However, in the grand scheme of things – also factoring in the lack of scalable alternate algorithms for this problem in the literature – we attain pretty promising results for *Rm-TWET* as well.

As before, the purpose of the figures presented under “**(TR – A)-BDS**” in

Columns 4–6 is to argue the value of the our approach for solving  $(\mathbf{TR} - \mathbf{A})$ . The number of instances not solved to within the targeted gap of 3% by  $(\mathbf{TR} - \mathbf{A})$ -BDS within 600 seconds is 159 out of a total of 720. The corresponding number for  $(\mathbf{TR} - \mathbf{A})$ -CPX is 252 with a time limit of 1800 seconds. Moreover, the median gap of 8.3% for those instances that could not be solved within the specified time limit by  $(\mathbf{TR} - \mathbf{A})$ -BDS stands in stark contrast to the corresponding gap of 100% for  $(\mathbf{TR} - \mathbf{A})$ -CPX. The respective average gaps are 12.6% and 80.2%. We reckon that  $(\mathbf{TR} - \mathbf{A})$ -BDS tackles the preemptive relaxation  $(\mathbf{TR} - \mathbf{A})$  of  $Rm$ -TWET successfully. In addition, observe that the monolithic formulation of  $(\mathbf{TR} - \mathbf{A})$  with 200 jobs and 5 machines grows too large for CPLEX, and even the root relaxation is not solved within the allotted time. Therefore, no results are reported for  $(\mathbf{TR} - \mathbf{A})$ -CPX for this instance size.

$(\mathbf{TR} - \mathbf{A})$ -BDS yields very good lower bounds for  $Rm$ -TWET. The average lower bound gap in Columns 7–9 is no more than 14.75% for all  $n, m$  combinations with an average of 9.02% across all instances. The gap is in excess of 15% for only 13% of the instances (93 instances).

The results on the optimality gaps of the non-preemptive solutions included under “Feasible Sol’n” in Table 2.4 certify  $(\mathbf{TR} - \mathbf{A})$ -BDS as a viable and scalable algorithm for solving large instances of  $Rm$ -TWET. As is the case with  $Rm$ -TWT, even the LP relaxation of  $(\mathbf{TI})$  is not solved within half an hour for instances with 100 or more jobs. Among the smaller 360 instances,  $(\mathbf{TI})$  beats  $(\mathbf{TR} - \mathbf{A})$ -BDS in 125 cases with an average improvement of 0.84%. For the other 235 instances,  $(\mathbf{TR} - \mathbf{A})$ -BDS outperforms  $(\mathbf{TI})$  by 40.17% on average. The optimality gap of the incumbent from  $(\mathbf{TI})$  is over 15% in 39% of these instances (142 instances) while  $(\mathbf{TR} - \mathbf{A})$ -BDS does always keep the gap below the same threshold with the exception of 5 instances. Even for the 360 larger instances with 100 or more jobs, the proposed Benders-type method finds a feasible solution for the original problem with an optimality gap less than 15% in 86% of the cases (310 instances). The behavior of  $(\mathbf{TR} - \mathbf{A})$ -BDS with respect to the varying TF and RDD levels in Table 2.4 is consistent with our observations for Table 2.3 in Section 2.5.1. The adverse effect of low TF and high RDD values on both the lower and upper bound quality persists with the same underlying reasons explained in Section 2.5.1.

**Table 2.4** Results for *Rm-TWET*.

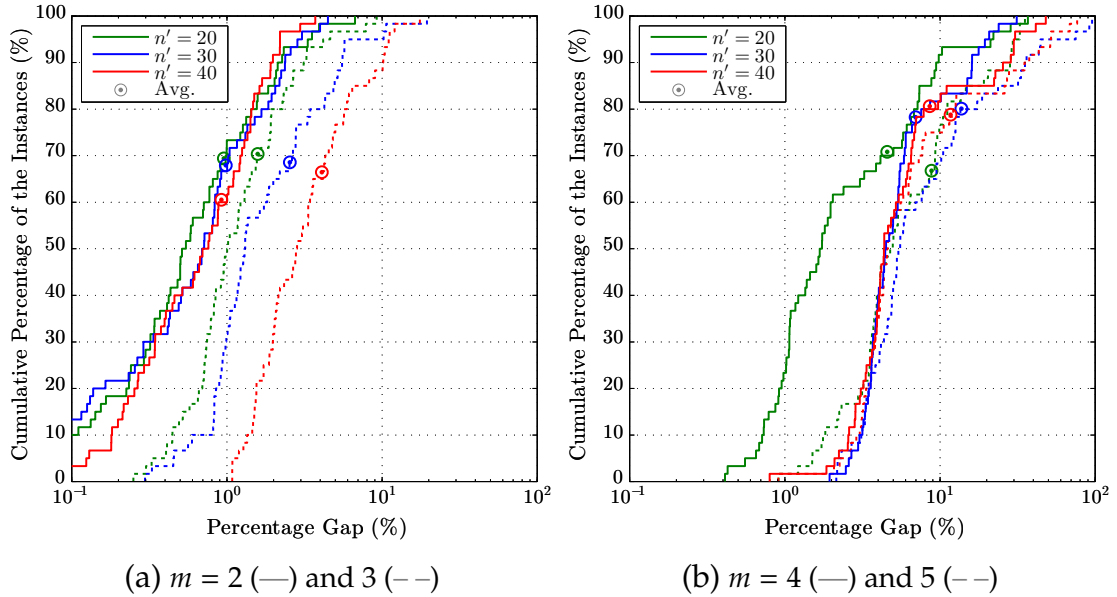
<i>n</i>	RDD		Percentage Gaps									(TR – A) - Solution Times					
	<i>m</i>	TF	(TR – A)-BDS			LB Quality			Feasible Sol'n			CPX			BDS		
			0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
40	2	0.4	2.7	2.9	4.4	4.7	6.9	15.1	0.8	0.9	3.2	15	13	21	16	22	454
		0.6	2.6	2.8	2.9	4.7	7.4	10.3	0.8	2.0	0.7	16	12	14	14	21	48
		0.8	2.0	2.3	2.8	3.1	4.6	5.7	1.0	0.8	0.6	17	15	14	10	10	17
		1.0	1.9	1.4	1.5	2.2	2.2	2.7	0.3	0.2	0.2	16	16	15	5	6	8
60	2	0.4	2.1	3.0	5.4	3.8	6.6	13.7	0.7	1.1	2.7	59	43	47	47	146	550
		0.6	2.2	2.7	3.0	3.9	6.8	10.0	0.7	1.7	2.6	62	48	45	41	52	249
		0.8	1.8	1.8	2.6	2.9	3.8	5.7	0.4	0.6	0.8	65	56	53	24	35	72
		1.0	1.6	0.6	1.7	1.8	1.1	2.8	0.2	0.1	0.3	69	71	70	14	23	26
80	3	0.4	2.6	3.0	10.1	5.3	8.1	17.4	1.5	1.3	2.3	121	219	384	17	207	601
		0.6	2.9	2.9	6.0	5.0	7.9	17.9	1.4	2.3	5.3	199	183	382	6	20	541
		0.8	2.0	2.6	2.9	3.4	5.7	7.0	0.9	0.8	1.1	256	216	211	3	9	18
		1.0	2.4	2.4	2.6	2.4	2.7	4.3	0.7	0.7	0.8	231	237	232	2	2	5
90	2	0.4	2.1	2.8	6.6	3.5	5.6	13.3	0.9	1.1	2.6	198	156	118	68	142	588
		0.6	2.0	2.7	3.1	3.1	5.4	7.9	0.5	1.5	1.7	194	140	143	47	74	328
		0.8	1.9	1.7	2.6	2.6	3.1	5.0	0.4	0.5	1.0	210	198	155	35	51	70
		1.0	1.9	1.0	1.0	2.0	1.5	1.9	0.3	0.2	0.3	227	224	204	19	33	39
100	4	0.4	2.9	5.9	31.2	5.8	12.1	50.7	2.8	4.9	20.6	433	647	1708	48	596	609
		0.6	2.7	3.6	20.3	6.4	9.7	35.6	2.8	4.6	11.5	604	611	1947	23	324	605
		0.8	2.4	2.8	3.0	4.2	6.2	8.2	1.1	1.9	1.5	628	526	644	17	31	426
		1.0	2.8	2.1	2.7	3.1	3.1	4.6	1.0	0.9	1.4	773	685	680	10	8	18
120	3	0.4	2.7	3.7	15.2	4.6	8.3	27.3	1.9	3.3	8.6	439	495	920	16	335	606
		0.6	2.1	2.8	6.6	4.0	6.8	15.7	1.2	3.0	5.3	721	595	757	13	120	605
		0.8	2.8	2.7	2.6	3.9	5.0	5.9	0.8	1.6	1.9	800	716	578	10	11	43
		1.0	2.7	2.4	2.2	2.7	3.4	3.4	1.0	0.9	1.0	739	691	669	8	10	11
150	5	0.4	3.1	9.6	34.6	6.1	16.9	57.9	4.2	10.5	30.9	1555	1805	1849	394	611	606
		0.6	2.9	6.5	22.2	6.1	14.1	41.5	4.5	10.0	22.9	1498	1670	1804	99	601	600
		0.8	2.6	3.0	6.2	4.7	6.5	12.0	3.4	5.1	7.4	1363	1111	1399	8	109	490
		1.0	2.5	1.9	2.8	3.0	3.1	5.1	1.4	2.1	3.5	1614	1480	1382	3	4	16
160	4	0.4	2.4	3.0	11.5	3.8	6.6	20.9	2.5	5.0	12.0	1331	1053	978	33	376	601
		0.6	1.9	2.9	5.0	3.2	6.5	12.8	2.0	4.9	9.2	1448	1098	1037	15	136	601
		0.8	2.7	2.2	2.8	3.6	4.1	5.5	1.5	2.9	3.9	1679	1416	1103	9	13	55
		1.0	2.7	2.4	2.2	2.9	3.0	3.2	1.3	2.0	1.8	1699	1726	1481	7	7	11
160	4	0.4	2.9	4.3	24.4	5.1	8.6	41.4	3.8	5.9	22.6	1710	1568	1819	64	601	601
		0.6	2.6	3.5	12.2	4.6	8.6	24.8	4.3	6.5	15.6	1805	1691	1789	19	311	601
		0.8	2.4	2.8	3.2	3.8	5.4	7.4	3.8	4.8	6.3	1812	1802	1692	9	56	244
		1.0	2.7	2.5	2.3	3.2	3.4	3.9	3.0	3.4	3.6	1812	1806	1803	5	7	10
160	5	0.4	2.9	8.3	30.4	4.9	14.8	66.9	4.9	14.8	66.0	1895	1865	2093	154	601	602
		0.6	2.9	4.8	17.3	5.2	10.8	36.0	5.2	10.8	36.0	1843	1986	1859	46	601	601
		0.8	2.4	2.9	3.8	4.0	5.7	8.3	4.0	5.7	8.3	1858	1858	1850	13	83	507
		1.0	2.5	1.9	2.1	3.0	2.8	3.6	3.0	2.8	3.6	1900	1822	1812	7	10	12
160	4	0.4	2.6	4.3	18.1	3.9	8.5	35.1	3.9	8.5	35.1	1880	1850	1819	78	489	601
		0.6	2.1	3.0	10.6	3.6	6.6	22.7	3.6	6.6	22.7	1876	1836	1880	23	200	601
		0.8	2.6	2.6	2.8	3.7	4.6	5.8	3.7	4.6	5.8	1907	1841	1843	15	41	138
		1.0	2.1	2.6	1.8	2.5	3.3	3.0	2.3	3.3	3.0	1886	1844	1836	10	11	18

Continued on next page...



Table 2.4 continued...

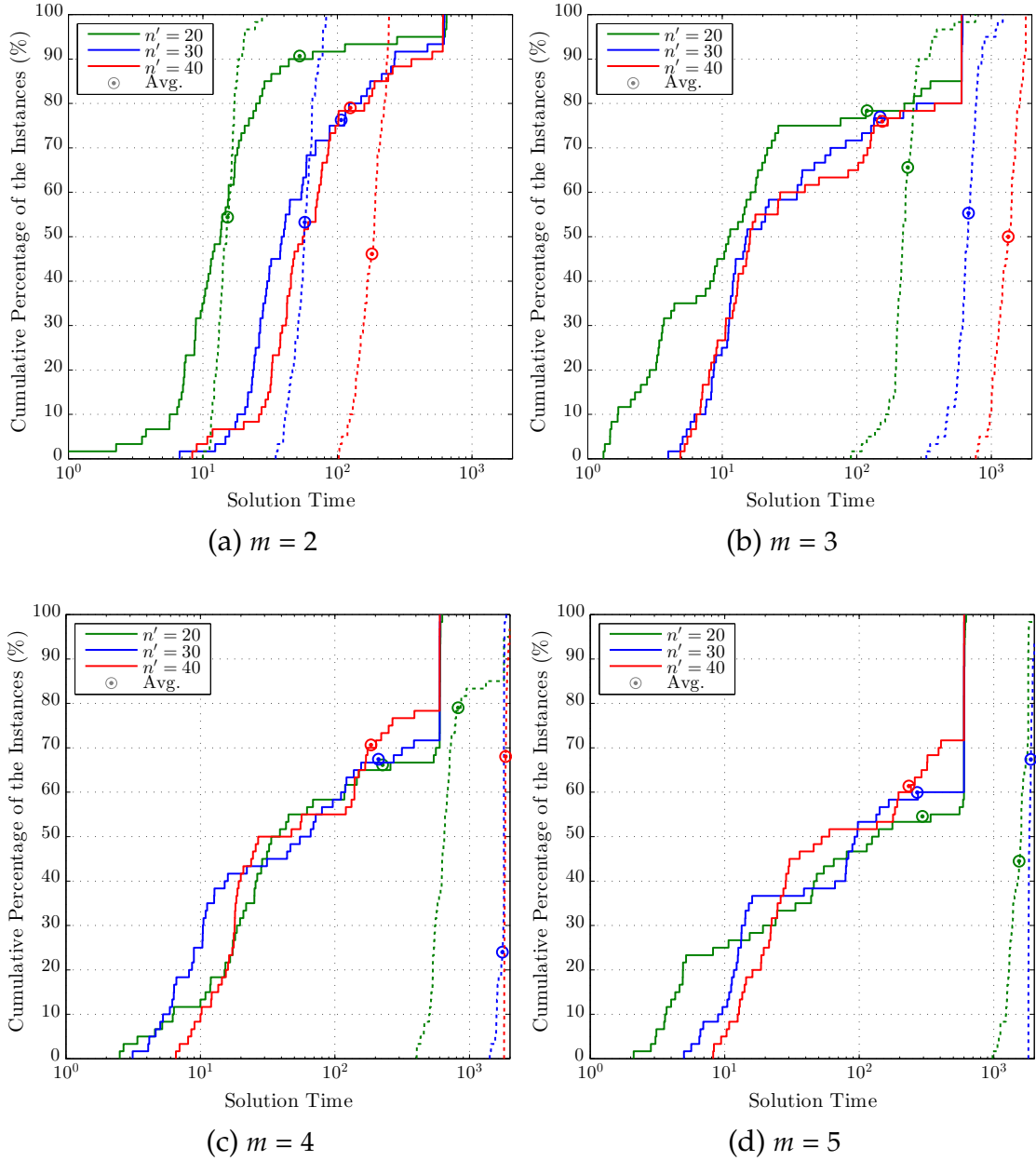
RDD		Percentage Gaps									(TR – A) - Solution Times						
		(TR – A)-BDS			LB Quality			Feasible Sol'n			CPX			BDS			
$n$	$m$	TF	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
200	5	0.4	2.9	6.9	28.3	4.4	11.9	58.4	4.4	11.9	58.4				127	601	601
		0.6	2.1	2.9	14.9	3.5	7.2	30.7	3.5	7.2	30.7				28	325	601
		0.8	2.0	2.8	3.0	3.0	4.7	6.4	3.0	4.7	6.4				22	81	378
		1.0	2.3	2.8	2.7	2.7	3.6	3.9	2.7	3.6	3.9				11	13	19



**Figure 2.3** The empirical distributions of the optimality gaps of the upper bounds by (TR – A)-BDS for  $Rm$ -TWET.

Figure 2.3 is the counterpart of Figure 2.1 in Section 2.5.1 where the empirical distributions of the optimality gaps of the feasible solutions associated with (TR – A)-BDS are plotted. As previously, (TR – A)-BDS generally exhibits a robust behavior with respect to varying values of  $n'$  for a fixed  $m$ . The problem gets more challenging with an increasing number of machines, and the percentage gaps associated with (TR – A)-BDS demonstrate a modest increase with increasing  $m$ . For instance, for 70% of the instances with 2, 3, 4, and 5 machines, the gaps are less than 2%, 5%, 7%, and 11%, respectively.

The performance patterns observed for  $Rm$ -TWT pretty much carry over to  $Rm$ -TWET as well. The solution times of (TR – A)-BDS are in general better than those of (TR – A)-CPX by a large margin. Based on the 399 instances that are solved by both methods within their respective time limits, the ratio of the solution time of (TR – A)-CPX to that of (TR – A)-BDS is 48.0 on average. Among



**Figure 2.4** The empirical distributions of the solution times of **(TR – A)-BDS** and **(TR – A)-CPX** for *Rm-TWET*.

these instances, only 52 of the relatively smaller instances with less than 90 jobs and generally large RDD values take longer for **(TR – A)-BDS**. As in Table 2.3 in Section 2.5.1, low TF and high RDD values result in tough instances to handle for **(TR – A)-BDS** while instances with tight due dates are solved extremely well.

The empirical distributions of the solution times of **(TR – A)-BDS** and **(TR – A)-CPX**, plotted with solid and dashed lines in Figure 2.4, respectively, reveal that the median solution times of **(TR – A)-BDS** are in the range from 11 to 125 seconds for all  $m, n'$  combinations. **(TR – A)-CPX** features a much less robust behavior

with a median solution time of 15 seconds for  $n' = 20$  and  $m = 2$  that quickly increases to 220, 649, and 1589 seconds for  $n' = 20$  and  $m = 3, 4, 5$ , respectively. Compared to those in Table 2.3 in Section 2.5.1, the computational effort expended is significantly more. To be specific, the median solution times of **(TR – A)-CPX** for the  $Rm$ -TWET instances with 2, 3, 4, and 5 machines are 5, 7, 7, and 4 times of those for the corresponding  $Rm$ -TWT instances, respectively. The respective ratios for **(TR – A)-BDS** are 11, 4, 6, and 8. The greater planning horizons in the formulations are a primary factor here in addition to the inherent difficulty of  $Rm$ -TWET over  $Rm$ -TWT. This difficulty is also reflected in the number of Benders cuts generated. **(TR – A)-BDS** needs to create 5.7 times more cuts for  $Rm$ -TWET compared to  $Rm$ -TWT, and the great majority of these cuts is not redundant. The median percentage of the active Benders cuts for the final node problem in the search tree is 95.5% with a corresponding average of 92.1%. Note that these numbers are higher than their counterparts for  $Rm$ -TWT.

The times expended to solve the single-machine problems for a given job partition – omitted from Table 2.4 for the sake of brevity – are more than satisfactory. The **SiPS/SiPSi** solver returns the optimal solution for a single-machine TWET problem in about 27, 110, and 305 milliseconds for instances with  $n' = 20, 30, 40$ , respectively. These numbers translate into 23 seconds on average to solve all single-machine problems to optimality for a five machine and 200 job instance with a maximum of 56 seconds. While these figures are greater than their counterparts for  $Rm$ -TWT, they still make up for a small part of the total solution time. The time spent for calculating the non-preemptive solutions accounts for only 12.5% of the total solution time on average. Finally, we note that **(TR – A)-BDS** identifies on average 5.4 times more job partitions per instance compared to **(TR – A)-CPX**, where the average number of partitions retrieved from the search tree of **(TR – A)-BDS** is 14.4. As we discussed in Section 2.5.1, this is a critical advantage that improves the quality of the best non-preemptive solution.

## CHAPTER 3

# AN EXACT EXTENDED FORMULATION FOR TOTAL WEIGHTED COMPLETION TIME

The plethora of research on  $\mathcal{NP}$ -hard parallel machine scheduling problems is focused on heuristics due to the theoretically and practically challenging nature of these problems. Only a handful of exact approaches are available in the literature, and most of these suffer from scalability issues. Moreover, the majority of the papers on the subject are restricted to the identical parallel machine scheduling environment. In this context, the main contribution of this chapter is to recognize and prove that a particular preemptive relaxation for the problem of minimizing the total weighted completion time (TWCT) on a set of unrelated parallel machines naturally admits a non-preemptive optimal solution and gives rise to an exact mixed integer linear programming (MIP) formulation of the problem. Furthermore, we exploit the structural properties of TWCT and attain a very fast and scalable exact Benders decomposition-based algorithm for solving this formulation. Computationally, our approach holds great promise and may even be embedded into iterative algorithms for more complex shop scheduling problems as instances with up to 1000 jobs and 8 machines are solved to optimality within a few seconds.

### 3.1 Introduction

Motivated by the practical and theoretical considerations outlined in Chapter 1, our primary objective in this chapter is to devise a scalable effective exact method for solving the problem of minimizing the weighted completion time on a bank of unrelated parallel machines. The objective is one of the most frequently studied fundamental scheduling objectives as it tends to minimize the cycle time of the tasks on the shop floor. Formally, we characterize this problem as  $Rm//\sum_j w_j C_j$  ( $Rm$ - $TWCT$ ), following the three field notation of [Graham et al. \(1979\)](#) in classifying scheduling problems. The notation  $Rm$  in the first field stands for a bank of  $m$  unrelated machines. The completion time of job  $j$  is represented by  $C_j$  and penalized at a rate of  $w_j$  per unit time. [Bruno et al. \(1974\)](#) prove that minimizing the weighted completion time on two identical parallel machines is  $\mathcal{NP}$ -hard which also renders  $Rm$ - $TWCT$   $\mathcal{NP}$ -hard.

Any algorithm for a parallel machine scheduling problem has two major components: the jobs are assigned to the machines, and an optimal schedule is determined for each of the machines given the objective function and the job-to-machine assignments. In other words, we may think of a parallel machine scheduling problem as a set partitioning problem where computing the cost of a partition requires solving  $m$  independent single-machine scheduling problems. Based on the  $\mathcal{NP}$ -hardness of the classical set partitioning problem – where each assignment decision is explicitly associated with a fixed cost –, we are already aware that determining the optimal job-to-machine assignments is a challenging task. However, for a parallel machine scheduling problem there may be a second layer of difficulty if the underlying single-machine scheduling problem is not polynomially solvable. This – for instance – is the setting for the problems of minimizing the total weighted tardiness (TWT) and total weighted earliness/tardiness (TWET) on  $m$  unrelated parallel machines studied in Chapter 2, where the corresponding single-machine problems are strongly  $\mathcal{NP}$ -hard. These two problems are referred to as  $Rm$ - $TWT$  and  $Rm$ - $TWET$  in the sequel, respectively. In contrast, in this chapter the cost of a given partition is calculated in polynomial time by applying the well-known weighted shortest processing time (WSPT) rule ([Smith, 1956](#)) to each of the  $m$  machines separately. This really is the key difference that

allows us to turn an MIP formulation that only yields lower bounds for  $Rm-TWT$  and  $Rm-TWET$  into an exact formulation for  $Rm-TWCT$ .

The work in Chapter 2 was motivated by the lack of strong lower bounds for parallel machine scheduling problems with additive objectives (van den Akker et al., 1999) and focused on the due date related objectives TWT and TWET. Given the two-stage structure inherent in parallel machine scheduling discussed above, the key idea in that chapter is to replace the non-preemptive scheduling decisions on a machine by a tight preemptive relaxation solved as a linear programming (LP) problem in an extended variable space. This in turn allows us to propose a preemptive relaxation for the original problems  $Rm-TWT$  and  $Rm-TWET$ , where the preemptive relaxation is formulated as an MIP amenable to a solution approach that relies on Benders decomposition (Benders, 1962). The job-to-machine assignment decisions are kept in the master problem, and the cost of these decisions is approximated by Benders cuts, generated by solving a separate LP problem for each machine. In this chapter, we prove that the same MIP is an exact formulation for  $Rm-TWCT$ . This in essence requires that the single-machine TWCT problem is solved to optimality as an LP problem – see Corollary 3.5. Moreover, by exploiting the structure of the TWCT objective we demonstrate that the Benders cuts are generated analytically without the need to invoke any LP algorithm. These results collectively provide us with a scalable and very effective exact algorithm for  $Rm-TWCT$ . To put it into perspective, we note that the best performing heuristic for  $Rm-TWCT$  to date (Rodriguez et al., 2013) runs with a time limit of  $n$  seconds, where  $n$  is the number of jobs, on instances with up to 1000 jobs and 50 machines. None of the previous studies on heuristics, e.g., (Vredeveld and Hurkens, 2002), (Lin et al., 2011), (Rodriguez et al., 2012), report results with more than 200 jobs. The computational results in Section 3.4 illustrate that our exact method solves instances with up to 8 machines and 1000 jobs to optimality within less than 12 seconds on average for any  $n$  and  $m$  combination in the indicated range. Instances with 16 and 30 machines are more time consuming, but we impose a time limit of 300 seconds, and instances not solved to optimality within the allotted time almost always terminate with incumbents within 1% of optimality.

In the next section, we review the related literature to position our work. In

Section 3.3, we give pointers to the existing monolithic mathematical programming models for  $Rm-TWCT$ , and then present our own formulation for  $Rm-TWCT$  and prove its correctness before introducing the solution algorithm based on Benders decomposition. The computational results in Section 3.4 attest to the efficacy of our approach.

## 3.2 Review of Related Literature

The early focus of the parallel machine scheduling literature is on the makespan and total (weighted) completion time objectives with an emphasis on the polynomially solvable problems and approximation algorithms for the makespan (Cheng and Sin, 1990). Pinedo (2008) provides an in-depth discussion of the polynomially solvable cases and the associated structural results of interest. More recent surveys on parallel machine scheduling include (Mokotoff, 2001) and (Blazewicz et al., 2007, Chapter 5). A detailed discussion of the parallel machine scheduling literature on additive due date related performance measures is presented in Section 2.2 of Chapter 2. Note that the performance figures presented in this section are obtained by their respective authors on different computing platforms.

The literature on  $Rm-TWCT$  can be categorized into three streams: (meta-)heuristics, approximation algorithms, and exact approaches. A good overview of the (meta-)heuristics is provided in (Li and Yang, 2009, Rodriguez et al., 2013). Approximation algorithms for  $Rm-TWCT$  rely on rounding the optimal solution of a linear or convex quadratic programming relaxation of the problem. We refer the interested reader to (Chekuri and Khanna, 2004), where the authors survey the approximation algorithms for minimizing the total weighted completion time in different machine environments, and to (Li and Yang, 2009). The literature on exact methods for  $TWCT$  in the parallel machine environment creates the context for our study, and we restrict our attention to these in the sequel. Interestingly, all exact algorithms for  $TWCT$  in the parallel machine environment are limited to identical machines up until 1999. These include the branch-and-bound (B&B) algorithms of Elmaghraby and Park (1974), Barnes and Brennan (1977), Sarin et al. (1988), Belouadah and Potts (1994), and the dynamic programming techniques of Lawler and Moore (1969), Lee and Uzsoy (1992). Unsurprisingly, these

papers have very limited success in solving instances of any meaningful size – clearly partly due to the lack of powerful computers back then. However, we observe that even more recent and modern implementations of B&B algorithms are hardly effective in practice for the total (weighted) completion time problems in the same machine environment. The B&B method of [Yalaoui and Chu \(2006\)](#) for  $Pm/r_j/\sum_j C_j$ , where  $Pm$  stands for the identical parallel machine environment and  $r_j$  in the second field indicates that jobs may have different ready times, handles at most 45 jobs for 5 and 10 machines and 120 jobs for 2 machines only with a 30 minute time limit. Another B&B algorithm is devised by [Nessah et al. \(2008\)](#) for  $Pm/r_j/\sum_j w_j C_j$  and solves instances with up to 60 jobs and 5 machines in one hour of CPU time. The first two B&B procedures for the non-identical parallel machine environment are due to [Azizoglu and Kirca \(1999a\)](#) and [Azizoglu and Kirca \(1999b\)](#). In their earlier work, the authors tackle the problems  $Pm//\sum w_j C_j$  and  $Qm//\sum w_j C_j$ , where  $Qm$  denotes the presence of  $m$  uniform parallel machines. In either case, their B&B algorithm does not scale beyond 3 machines and 25 jobs in 10 and 15 minutes of CPU time for  $Pm//\sum w_j C_j$  and  $Qm//\sum w_j C_j$ , respectively. The first exact solution procedure for our problem  $Rm-TWCT$  appears in ([Azizoglu and Kirca, 1999b](#)). The B&B method of the authors incorporates structural dominance properties to exclude unpromising nodes from consideration. The lower bounding mechanism is based on solving an assignment problem and requires calculating a lower bound on the completion time of each job at each position on each machine. The computational results demonstrate that instances larger than 2 machines and 25 jobs or 3 machines and 20 jobs are beyond the approach with a 15 minute time limit.

From the discussion above, it is evident that the scalability of B&B methods for parallel machine (weighted) completion time problems is highly doubtful. It turns out that compared to custom B&B procedures, exact solution methods that rely on mathematical programming based decomposition techniques are far more promising for parallel machine scheduling problems with additive objective functions. This is true for both the total (weighted) completion time and the due date related performance measures. The underlying reason for this phenomenon is rooted in the tight lower bounds attained by good mathematical programming formulations of parallel machine scheduling problems. Two prime examples are



presented by [Chen and Powell \(1999b\)](#) and [van den Akker et al. \(1999\)](#). In both of these papers, a general parallel machine scheduling problem with an additive objective function of the job completion times is formulated as a set partitioning problem with exponentially many variables. Each variable (column) in the formulation corresponds to a feasible machine schedule. The branch-and-price algorithms of both sets of authors apply column generation (CG) to the node LP problems due to the huge number of feasible machine schedules, and they mainly differ in their branching schemes. The root relaxation often yields an integer optimal solution, and even if this property does not hold for a particular instance, in a vast majority of cases only a few nodes of the search tree need to be explored until the integer optimal solution is identified. Thus, both studies attribute their relative computational success to the quality of the LP relaxation of their set partitioning formulations. Another common trait of both branch-and-price algorithms is that decreasing the number of machines for a fixed number of jobs has a detrimental effect on the computational performance. [Chen and Powell \(1999b\)](#) apply their solution method to *Rm-TWCT* among others and report results with up to 100 jobs and 20 machines. Their average CPU time for instances with 100 jobs degrades from 363 seconds with 20 machines to 2051 seconds with 8 machines. [van den Akker et al. \(1999\)](#) report computational experience only with  $Pm // \sum_j w_j C_j$ , and their results are similar. Aggregated over all three instance classes the authors consider, the average solution time for instances with 100 jobs and 10 machines is 1512 seconds. Finally, the best contender in the literature as an exact solution method for *Rm-TWCT* turns out to be solving the convex quadratic integer programming (CQIP) formulation of [Skutella \(2001\)](#) who propose this formulation – presented in Section 3.3 – as a means of developing an approximation algorithm for *Rm-TWCT* with a performance guarantee of  $3/2$ . Later, [Plateau and Rios-Solis \(2010\)](#) perform an experimental study on this formulation and find out that it attains the best computational results for *Rm-TWCT* to date. We benchmark the performance of our new MIP formulation solved by Benders decomposition against that of this CQIP formulation in Section 3.4.

The review of the related literature reveals that developing a scalable exact algorithm for *Rm-TWCT* is still an open research question – in particular if the ratio of the number of jobs to the number of machines is not small. This observation

and the success of the mathematical programming based solution methods for parallel machine scheduling problems detailed above motivates the work in this chapter. In this sense we follow suit with [Chen and Powell \(1999b\)](#), [van den Akker et al. \(1999\)](#) and apply a decomposition approach to a mathematical programming formulation that is demonstrated to provide tight lower bounds for the TWT and TWET objectives in the unrelated parallel machine environment in Section 2.5 of Chapter 2.

### 3.3 Formulation and Solution Approach

In our problem *Rm-TWCT*, a set of  $n$  jobs are ready at time zero to be processed on a bank of  $m$  unrelated parallel machines. Each job is required to receive service from exactly one machine. All machines are available continuously from time zero onward, and a machine can execute at most one job at a time. Without loss of generality, any machine can process any job, and if job  $j$  is performed on machine  $k$ , then it stays on the machine for an integer duration of  $p_{jk}$  time units without interruption – preemption is not allowed. As introduced in Section 3.1, the objective is to minimize  $\sum_j w_j C_j$ , where  $w_j$  and  $C_j$  indicate the unit completion time penalty and the completion time associated with job  $j$ , respectively.

An overview of the exact monolithic mathematical programming formulations available for *Rm-TWCT* is provided in [\(Li and Yang, 2009\)](#) and [\(Unlu and Mason, 2010\)](#), and we can infer that the time-indexed formulation – initially introduced in a seminal paper by [Dyer and Wolsey \(1990\)](#) in the single-machine context – stands out amongst the MIP formulations. While time-indexed formulations give rise to tight LP relaxations, their pseudo-polynomial size turns into a major drawback for instances with long processing times. [Unlu and Mason \(2010\)](#) report that the time-indexed formulation outperforms the other MIP formulations on  $P2/\sum_j w_j C_j$  and  $P3/\sum_j w_j C_j$  instances with up to 100 jobs and a time limit of one hour, but does not scale beyond – in particular if the maximum processing time is increased to 100 from 20. The findings of a recent study by [Berghman et al. \(2014\)](#) provide further evidence in this direction. The authors consider general cost functions and improve the original time-indexed formulation by preprocessing and new valid inequalities. The results demonstrate no real benefit from the proposed

techniques, and many of the 200-job instances with 2 machines and a maximum processing time of 20 remain unsolved in one hour. Similar observations on linear formulations lead [Rodriguez et al. \(2013\)](#) to consider a quadratic integer programming formulation for evaluating the quality of their heuristics, and we follow suit with them in this sense. As mentioned at the end of Section 3.1, the CQIP formulation proposed by [Skutella \(2001\)](#) and evaluated empirically by [Plateau and Rios-Solis \(2010\)](#) exhibits the best computational performance on  $Rm$ -TWCT to date and is stated below. We benchmark against this formulation in Section 3.4.

$$(CQ) \quad \text{minimize} \quad \sum_{j=1}^n \sum_{k=1}^m \left( \frac{1}{2} w_j p_{jk} (y_{jk} + y_{jk}^2) + \sum_{i <_k j} w_j p_{ik} y_{ik} y_{jk} \right) \quad (3.1)$$

$$\text{subject to} \quad \sum_{k=1}^m y_{jk} = 1, \quad j = 1, \dots, n, \quad (3.2)$$

$$y_{jk} \in \{0, 1\}, \quad j = 1, \dots, n, k = 1, \dots, m. \quad (3.3)$$

In (CQ), setting the value of the binary variable  $y_{jk}$  to one implies that job  $j$  is to be processed on machine  $k$ . Each job is assigned to exactly one machine by the job partitioning constraints (3.2). The notation  $i <_k j$  implies that either  $\frac{w_i}{p_{ik}} > \frac{w_j}{p_{jk}}$  or  $\frac{w_i}{p_{ik}} = \frac{w_j}{p_{jk}}$  and  $i < j$  following the WSPT order on machine  $k$ . (CQ) relies on the basic observation  $C_j = \sum_{k=1}^m y_{jk} (p_{jk} + \sum_{i <_k j} p_{ik} y_{ik})$  and a convexification of the resulting objective function.

The foundation of our exact formulation for  $Rm$ -TWCT resides in a class of tight lower bounds initially developed for the single-machine weighted tardiness and weighted earliness/tardiness scheduling problems by [Sourd and Kedad-Sidhoum \(2003\)](#), [Bülbül et al. \(2007\)](#), [Pan and Shi \(2007\)](#), and [Şen and Bülbül \(2012\)](#). The fundamental idea in this body of work is to allow jobs to be preempted at integer points in time and to penalize the completion time of each unit-length job. The problem of determining the best schedule with this preemptive scheme is then formulated as an assignment or a transportation problem, in which a job  $j$  with an integer processing time  $p_j$  is allocated a total of  $p_j$  unit-length intervals in the planning horizon and the machine executes no more than a single unit-length job at a time. The size of the formulation is pseudo-polynomial because the length of

the planning horizon is determined by the sum of the processing times and the magnitude of the due dates. Still, the existence of very effective algorithms for the assignment and transportation problems ensures the viability of this lower bounding method. Later, [Kedad-Sidhoum et al. \(2008\)](#) extend this lower bounding approach to the identical parallel machine environment by reducing  $m$  identical parallel machines to a single-machine with a capacity of executing  $m$  unit-length jobs simultaneously. However, the optimal solution of the preemptive relaxation formulated as a transportation problem in the identical parallel machine environment lacks a crucial piece of information. The unit-length jobs of a given job cannot overlap in time, but they can be performed on different machines. The absence of an explicit assignment of the jobs to the machines in this relaxation is a major hurdle to the design of optimal or heuristic algorithms that would rely on the job-to-machine assignments for branching decisions or constructing non-preemptive individual machine schedules. Moreover, applying the machine capacity aggregation technique of [Kedad-Sidhoum et al. \(2008\)](#) would require further loss of information in the case of unrelated parallel machines – by setting  $p_j = \min_k p_{jk}$  for job  $j$  –, and the issue of inferring correct job-to-machine assignment decisions from the optimal solution of the preemptive relaxation would only be exacerbated. Motivated by these observations, we propose the formulation (TR – A) in Section 2.3 of Chapter 2 which mandates that all unit-length jobs of a given job are executed on the same machine. This formulation is reintroduced below for completeness.

$$(TR - A) \quad \text{minimize} \quad \sum_{j=1}^n \sum_{k=1}^m \sum_{t=1}^H c_{jkt} x_{jkt} \quad (3.4)$$

$$\text{subject to} \quad \sum_{t=1}^H x_{jkt} = p_{jk} y_{jk}, \quad j = 1, \dots, n, k = 1, \dots, m, \quad (3.5)$$

$$\sum_{j=1}^n x_{jkt} \leq 1, \quad k = 1, \dots, m, t = 1, \dots, H, \quad (3.6)$$

$$\sum_{k=1}^m y_{jk} = 1, \quad j = 1, \dots, n, \quad (3.7)$$

$$x_{jkt} \geq 0, \quad j = 1, \dots, n, k = 1, \dots, m, t = 1, \dots, H, \quad (3.8)$$

$$y_{jk} \in \{0, 1\}, \quad j = 1, \dots, n, k = 1, \dots, m. \quad (3.9)$$

In the model **(TR – A)**, the time period  $t$  represents the time interval  $(t - 1, t]$ , and the variable  $x_{jkt}$  is set to one at a cost of  $c_{jkt}$  if a unit-length job of job  $j$  is performed on machine  $k$  in time period  $t$ . The machine capacity constraints (3.6) prescribe that no more than one unit-length job is in process in period  $t$  on machine  $k$ . The constraints (3.5) ensure that all unit-length jobs of job  $j$  are carried out on the same machine because exactly one of the binary job-to-machine assignment variables  $y_{jk}$ ,  $k = 1, \dots, m$ , is set to one due to the job partitioning constraints (3.7). The end of the planning horizon  $H = \left\lceil \sum_{j=1}^n \max_k (p_{jk}) / m \right\rceil + p_{\max}$  is valid because there exists an optimal solution of the non-preemptive problem *Rm-TWCT* so that all jobs are brought to completion at or before  $H$ . See the explanation below Equation (2.5) in Section 2.3 of Chapter 2 for the details of determining the appropriate value of  $H$ . Note that  $p_{\max} = \max_{j,k} (p_{jk})$  may be omitted from the expression for  $H$  in the case of a single-machine.

The fundamental difference of **(TR – A)** compared to the earlier work in the domain of preemptive relaxations discussed above lies in the constraints (3.5). While these constraints remove the drawback of having the unit-length jobs of a given job distributed over multiple machines, they also destroy the desirable polyhedral structure, and we no longer have a transportation problem on our hands. It turns out that **(TR – A)** is an MIP formulation of pseudo-polynomial size, which grows very quickly with increasing  $m$ ,  $n$ , and long processing times. The key to solving this formulation effectively is to recognize that **(TR – A)** decomposes into into  $m$  independent transportation problems for any fixed assignment of the jobs to the machines. This observation renders any integrality restrictions on the variables  $x_{jkt}$  redundant – see (3.8) – and suggests a Benders decomposition algorithm (Benders, 1962) for tackling **(TR – A)**. The existence of powerful LP engines that can solve very large transportation problem instances in very short times and a fast custom procedure to strengthen the Benders cuts enables us to obtain tight lower bounds for *Rm-TWT* and *Rm-TWET* in Chapter 2 by solving **(TR – A)** to (near-)optimality. Our main contribution over Chapter 2 in this chapter is to prove that **(TR – A)** with the objective coefficients to be discussed next yields an exact formulation for TWCT. Moreover, we identify the analytic closed form of the optimal solutions of the Benders subproblems and generate cuts without the need of invoking an LP solver as was the case in Chapter 2. Coupling this with

further enhancements attained in the Benders cut strengthening procedure results in a scalable and very fast optimal solution technique for  $Rm$ -TWCT.

One issue that deserves special attention is the choice of the objective coefficients in  $(\mathbf{TR} - \mathbf{A})$ . In the context of the preemptive relaxations developed previously for earliness/tardiness scheduling problems, the cost coefficients must satisfy the sufficient condition (3.10) below, where  $f_j(t)$  is the cost incurred by job  $j$  in the original non-preemptive problem for completing at time  $t$ .

$$\sum_{s=t-p_{jk}+1}^t c_{jks} \leq f_j(t) \quad j = 1, \dots, n, k = 1, \dots, m, t = p_{jk}, \dots, H. \quad (3.10)$$

This condition states that the total cost accumulated by all  $p_{jk}$  unit-length jobs of job  $j$  on machine  $k$  in a non-preemptive solution of the preemptive relaxation does not exceed the cost that job  $j$  would incur in the original non-preemptive problem with the same completion time. Thus, the optimal objective value of the preemptive relaxation is guaranteed to be a lower bound on that of the corresponding non-preemptive problem.

An infinite number of cost coefficient vectors satisfy (3.10). The particular choice determines the strength of the lower bound and the empirical performance, and the existing papers in the literature differ from each other in this respect. For an in-depth discussion on this subject and the properties of alternate cost coefficients, the reader is referred to (Pan and Shi, 2007) and Section 2.3 of Chapter 2. In this chapter, we directly employ the cost coefficients used in Chapter 2 for  $Rm$ -TWT and  $Rm$ -TWET by setting all due dates equal to zero – both problems reduce to  $Rm$ -TWCT with zero due dates:

$$c_{jkt} = \frac{w_j}{p_{jk}} \left( t + \frac{p_{jk}}{2} - \frac{1}{2} \right), \quad j = 1, \dots, n, k = 1, \dots, m, t = 1, \dots, H. \quad (3.11)$$

Consequently, Proposition 3.1 presented next follows as a direct corollary of Proposition 2.1 given in Chapter 2.

**Proposition 3.1.** *The optimal objective function value of  $(\mathbf{TR} - \mathbf{A})$  with the cost coefficients given in (3.11) is a lower bound on the optimal objective function value of the original non-preemptive problem  $Rm$ -TWCT.*

In the rest of the chapter, any reference to  $(\mathbf{TR} - \mathbf{A})$  employs the set of cost

coefficients (3.11). Two further intermediate results proven next and Proposition 3.1 collectively yield our main result formalized in Theorem 3.4. In the sequel, a non-preemptive feasible solution of  $(\mathbf{TR} - \mathbf{A})$  refers to a feasible solution of  $(\mathbf{TR} - \mathbf{A})$  in which all unit-length jobs of any job are processed in consecutive periods.

**Lemma 3.2.** *The cost charged against any non-preemptive feasible solution of  $(\mathbf{TR} - \mathbf{A})$  is identical to the cost incurred by this schedule in the original non-preemptive problem Rm-TWCT.*

*Proof.* The proof follows from a more general argument in (Bülbül et al., 2007, Theorem 2.2). The relationship below, where job  $j$  is assigned to  $p_{jk}$  consecutive time periods from  $C_j - p_{jk} + 1$  to  $C_j$  on machine  $k$  holds for all jobs. This completes the proof.

$$\begin{aligned} \sum_{t=C_j-p_{jk}+1}^{C_j} c_{jkt} &= \frac{w_j}{p_{jk}} \sum_{t=C_j-p_{jk}+1}^{C_j} \left( t + \frac{p_{jk}}{2} - \frac{1}{2} \right) \\ &= \frac{w_j}{p_{jk}} \left( p_{jk}(C_j - p_{jk}) + \frac{p_{jk}(p_{jk} + 1)}{2} + \frac{p_{jk}(p_{jk} - 1)}{2} \right) = w_j C_j \end{aligned}$$

□

**Proposition 3.3.** *There exists a non-preemptive optimal solution of  $(\mathbf{TR} - \mathbf{A})$ . Furthermore, in this optimal solution the jobs assigned to each machine are sequenced in the WSPT order.*

*Proof.* For any given fixed job partition  $\bar{y}$ ,  $(\mathbf{TR} - \mathbf{A})$  decomposes into  $m$  independent single-machine transportation problems. Therefore, the key to this proof is to show that the individual machine schedules constructed by  $(\mathbf{TR} - \mathbf{A})$  are non-preemptive and follow the WSPT order. To this end, it is sufficient to restrict our attention to one arbitrary machine  $k$ . Without loss of generality, we assume that a subset of jobs  $J_k$  with  $|J_k| = n_k$  are assigned to machine  $k$  in an optimal solution of  $(\mathbf{TR} - \mathbf{A})$  and that these jobs are re-indexed in the WSPT order; that is,  $\frac{w_1}{p_{1k}} \geq \frac{w_2}{p_{2k}} \geq \dots \geq \frac{w_{n_k}}{p_{n_k k}}$ . The total processing time on machine  $k$  is represented by  $P_k = \sum_{j \in J_k} p_{jk}$ . In the following, we prove that in the optimal schedule of machine  $k$ , the first  $p_{1k}$  positions are occupied by the unit-length jobs of job 1, and these are followed by  $p_{2k}$  unit-length jobs of job 2, etc.

We first restate the problem of finding the optimal (possibly preemptive) schedule of the set of jobs  $J_k$  on machine  $k$  as an assignment problem **(AP)** – a special case of the transportation problem:

$$\text{(AP)} \quad \text{minimize} \quad \sum_{i=1}^{P_k} \sum_{t=1}^{P_k} c'_{it} \delta_{it} \quad (3.12)$$

$$\sum_{t=1}^{P_k} \delta_{it} = 1, \quad i = 1, \dots, P_k, \quad (3.13)$$

$$\sum_{i=1}^{P_k} \delta_{it} = 1, \quad t = 1, \dots, P_k, \quad (3.14)$$

$$\delta_{it} \in \{0, 1\}, \quad i = 1, \dots, P_k, \quad t = 1, \dots, P_k. \quad (3.15)$$

The formulation **(AP)** decouples the unit-length jobs of a given job and regards them as independent tasks. The first  $p_{1k}$  tasks belong to job 1, the next  $p_{2k}$  tasks are associated with job 2, and so on. The original job associated with task  $i$  is denoted by  $j(i)$ . The binary variable  $\delta_{it}$  takes on the value one at a cost of  $c'_{it} = c_{j(i)kt}$  – as defined in (3.11) – if task  $i$  is processed in period  $t$ . The constraints (3.13)-(3.14) mandate that each task is assigned to one period and vice versa, respectively.

The cost coefficient matrix  $C' = (c'_{it})$  of **(AP)** turns out to be a Monge matrix – it fulfills a very special property known as the Monge property (Burkard et al., 2009, Definition 5.5):

$$c'_{i_1 t_1} + c'_{i_2 t_2} \leq c'_{i_1 t_2} + c'_{i_2 t_1}, \quad 1 \leq i_1 < i_2 \leq P_k, \quad 1 \leq t_1 < t_2 \leq P_k. \quad (3.16)$$

To recognize this, we note that  $c'_{it} = c_{j(i)kt} = \frac{w_{j(i)k}}{p_{j(i)k}} \left( t + \frac{p_{j(i)k}}{2} - \frac{1}{2} \right)$  and verify (3.16) for any  $i_1 < i_2$  and  $t_1 < t_2$ :

$$\begin{aligned} & \frac{w_{j(i_1)k}}{p_{j(i_1)k}} \left( t_1 + \frac{p_{j(i_1)k}}{2} - \frac{1}{2} \right) + \frac{w_{j(i_2)k}}{p_{j(i_2)k}} \left( t_2 + \frac{p_{j(i_2)k}}{2} - \frac{1}{2} \right) \\ & \leq \frac{w_{j(i_1)k}}{p_{j(i_1)k}} \left( t_2 + \frac{p_{j(i_1)k}}{2} - \frac{1}{2} \right) + \frac{w_{j(i_2)k}}{p_{j(i_2)k}} \left( t_1 + \frac{p_{j(i_2)k}}{2} - \frac{1}{2} \right) \\ \Leftrightarrow & \frac{w_{j(i_1)k}}{p_{j(i_1)k}} t_1 + \frac{w_{j(i_2)k}}{p_{j(i_2)k}} t_2 \leq \frac{w_{j(i_1)k}}{p_{j(i_1)k}} t_2 + \frac{w_{j(i_2)k}}{p_{j(i_2)k}} t_1 \\ \Leftrightarrow & \left( \frac{w_{j(i_1)k}}{p_{j(i_1)k}} - \frac{w_{j(i_2)k}}{p_{j(i_2)k}} \right) t_1 \leq \left( \frac{w_{j(i_1)k}}{p_{j(i_1)k}} - \frac{w_{j(i_2)k}}{p_{j(i_2)k}} \right) t_2. \end{aligned} \quad (3.17)$$



The final inequality (3.17) holds because  $t_1 < t_2$  and  $i_1 < i_2$  implies  $\frac{w_{j(i_1)}}{p_{j(i_1)k}} \geq \frac{w_{j(i_2)}}{p_{j(i_2)k}}$ .

Assignment problems with Monge cost coefficient matrices exhibit a very simple optimal solution: they are solved by the identical permutation (Burkard et al., 2009, Proposition 5.7). Stated in the context of our problem, executing task  $i$  in period  $i$  solves (AP) optimally. This optimal solution does clearly correspond to a non-preemptive schedule on machine  $k$ , in which the jobs are sequenced in the WSPT order and delivers the desired result for (TR – A).  $\square$

**Theorem 3.4.** (TR – A) is an exact formulation for Rm-TWCT.

*Proof.* By Proposition 3.3, we can identify a non-preemptive optimal solution  $S^*$  of (TR – A). The associated objective value is a lower bound on the optimal objective value of Rm-TWCT based on Proposition 3.1. Moreover,  $S^*$  is also feasible for Rm-TWCT, and Lemma 3.2 assures that the cost it incurs with respect to Rm-TWCT is identical to the optimal objective value of (TR – A). Therefore,  $S^*$  must be an optimal schedule for Rm-TWCT.  $\square$

(TR – A) is thus an exact extended formulation for Rm-TWCT obtained via variable splitting because the completion time  $C_j$  of job  $j$  can be expressed as  $C_j \geq tx_{jkt}, j = 1, \dots, n, k = 1, \dots, m, t = 1, \dots, H$ .

The corollary below follows from Theorem 3.4 and suggests an LP-based alternative for solving the non-preemptive single-machine TWCT problem.

**Corollary 3.5.** The non-preemptive single-machine total weighted completion time problem is equivalent to a transportation problem of pseudo-polynomial size.

*Proof.* Theorem 3.4 assures that we can solve the single-machine TWCT problem to optimality by setting  $m = 1$  in (TR – A). However, in this case, the formulation is simplified by dropping the binary variables  $y_{jk}$  from the formulation along with the constraints (3.7). The resulting model is a transportation problem with  $n$  source nodes and  $\sum_j p_{j1}$  sink nodes, where the objective coefficients are defined by (3.11). The size of the transportation problem is pseudo-polynomial because the number of sink nodes depends on the magnitude of the processing times.  $\square$

The primal solution of the transportation problem entails assigning the values of the variables  $x_{j1t}, j = 1, \dots, n, t = 1, \dots, H$ , as prescribed by the WSPT order of

the jobs, and the closed form of the dual solution is specified in the next section as part of our Benders decomposition scheme – see (3.28). In either case, the solution procedure is very fast in practice; however, there is no way of getting around the theoretical pseudo-polynomial complexity because of the number of value assignments required.

The result in Corollary 3.5 was actually discovered previously in the context of the relaxations of the single-machine TWCT problem with release dates – the problem  $1/r_j/\sum_j w_j C_j$ . Dyer and Wolsey (1990) explore and compare the strengths of various relaxations of  $1/r_j/\sum_j w_j C_j$ . Their weaker time-indexed formulation (D) (Dyer and Wolsey, 1990, Section 5) boils down to (TR – A) with  $m = 1$  after some simple manipulation. The authors point out that the optimal objective value of this preemptive time-indexed formulation can be computed in  $O(n \log n)$  time based on a lower bounding algorithm proposed in (Posner, 1985) for the single-machine TWCT problem with deadlines. A discussion of the same transportation problem as a relaxation for  $1/r_j/\sum_j w_j C_j$  is also presented by Goemans et al. (2002) who design approximation algorithms for this problem. Thus, in a sense Corollary 3.5 is a unifying result. It is obtained by studying a special case of the preemptive time-indexed formulations of earliness/tardiness scheduling problems and offers a new perspective on an already known result in different contexts. However, from our point of view the primary significance of being able to solve the single-machine TWCT problem as a transportation problem derives from the valuable dual information extracted from the optimal LP solution. In decomposition algorithms for complex scheduling problems with a single-machine TWCT component, one may opt for solving the subproblems as an LP problem in pseudo-polynomial time for the sake of this dual information – as is the case in this chapter. We are not aware of any other paper in the literature which adopts a similar approach.

### 3.3.1 Benders Decomposition

As alluded to several times up to this point, (TR – A) exhibits a decomposable structure that lends itself to a fast solution algorithm based on Benders decomposition. In this section, we formalize this discussion, and our presentation of the

Benders master and subproblems follows closely that in Section 2.4 of Chapter 2. Then, we prove that the optimal solutions of the Benders subproblems in this chapter can be computed analytically in closed form – without having to resort to a generic LP or transportation problem solver as was the case in Chapter 2. Finally, we enhance the cut strengthening procedure given in Section 2.4.1 of Chapter 2 by exploiting the special structure of the optimal solutions of the Benders subproblems. The ability to solve the subproblems very quickly combined with the improved cut strengthening procedure gives our Benders decomposition algorithm an additional edge and allows us to solve instances with up to 1000 jobs and 30 machines exactly in short computational times as demonstrated in the next section.

(**TR – A**) decomposes into  $m$  independent transportation problems for any fixed partition of the jobs to the machines as specified by the values of the binary variables  $y_{jk}, j = 1, \dots, n, k = 1, \dots, m$ . Thus, for any given fixed  $\bar{\mathbf{y}}$  satisfying (3.7), (**TR – A**) is reformulated via the Benders decomposition principle by replacing the right hand side of the set of constraints (3.5) by  $p_{jk}\bar{y}_{jk}$  and removing the set of constraints (3.7) and (3.9) from the model. The resulting LP problem is referred to as (**TR – A**( $\bar{\mathbf{y}}$ )), and the dual variables associated with the set of constraints (3.5) and (3.6) are denoted by  $u_{jk}, j = 1, \dots, n, k = 1, \dots, m$ , and  $v_{kt}, k = 1, \dots, m, t = 1, \dots, H$ , respectively. The formulation below is then the dual of of (**TR – A**( $\bar{\mathbf{y}}$ )) and exposes the decomposition into  $m$  independent transportation problems:

$$z(\bar{\mathbf{y}}) = \sum_{k=1}^m z_k(\bar{\mathbf{y}}), \quad (3.18)$$

where

$$\left(\mathbf{DS}_k - \mathbf{F}\right) \quad z_k(\bar{\mathbf{y}}) = \text{maximize} \quad \sum_{j=1}^n p_{jk}\bar{y}_{jk}u_{jk} + \sum_{t=1}^H v_{kt} \quad (3.19)$$

$$\text{subject to} \quad u_{jk} + v_{kt} \leq c_{jkt}, \quad j = 1, \dots, n, t = 1, \dots, H, \quad (3.20)$$

$$v_{kt} \leq 0, \quad t = 1, \dots, H, \quad (3.21)$$

is the dual of the transportation problem (**TR** $_k$ ) for machine  $k$ . Adopting the

common terminology for Benders decomposition,  $(\mathbf{TR}_k)$  and  $(\mathbf{DS}_k - \mathbf{F})$  are also referred to as the *cut generation subproblem* and the *dual slave problem* for machine  $k$ , respectively, in the following discussion.

For each candidate  $\bar{y}$ ,  $(\mathbf{DS}_k - \mathbf{F})$ ,  $k = 1, \dots, m$ , provide a pair of dual vectors  $\bar{u}, \bar{v}$  so that the sum of the optimal objective function values of  $(\mathbf{DS}_k - \mathbf{F})$ ,  $k = 1, \dots, m$ , is equal to the cost of the best solution of  $(\mathbf{TR} - \mathbf{A})$  that can be attained from the job partition  $\bar{y}$ . Consequently, a Benders optimality cut of the form

$$\eta \geq \sum_{k=1}^m \left( \sum_{j=1}^n p_{jk} \bar{u}_{jk} y_{jk} + \sum_{t=1}^H \bar{v}_{kt} \right) \quad (3.22)$$

removes  $\bar{y}$  from further consideration, where  $\eta$  represents a lower bound on the optimal objective value of  $(\mathbf{TR} - \mathbf{A})$ . Moreover, note that  $(\mathbf{TR} - \mathbf{A}(\bar{y}))$  is always feasible and only optimality cuts need to be generated. In the resulting relaxed Benders master problem **(RMP)** presented below, the cuts (3.22) appear in the disaggregated form (3.25) because this so-called *multi-cut* version proved superior in our preliminary computational experiments.

$$\text{(RMP) minimize } \sum_{k=1}^m \eta_k \quad (3.23)$$

$$\text{subject to } \sum_{k=1}^m y_{jk} = 1, \quad j = 1, \dots, n, \quad (3.24)$$

$$\eta_k \geq \sum_{j=1}^n p_{jk} \bar{u}_{jk}^c y_{jk} + \sum_{t=1}^H \bar{v}_{kt}^c, \quad k = 1, \dots, m, \quad c = 1, \dots, C, \quad (3.25)$$

$$y_{jk} \in \{0, 1\}, \quad j = 1, \dots, n, \quad k = 1, \dots, m. \quad (3.26)$$

In **(RMP)**, the number of times the dual slave problems  $(\mathbf{DS}_k - \mathbf{F})$ ,  $k = 1, \dots, m$ , have been solved so far is designated by  $C$ , and the superscript  $c$  in  $\bar{u}_{jk}^c$  and  $\bar{v}_{kt}^c$  indicates that these optimal values of the dual variables are obtained in iteration  $c$  of the cut generation. The auxiliary variable  $\eta_k$  approximates the total cost charged against the jobs performed on machine  $k$  from below, and the objective function value  $\sum_{k=1}^m \eta_k$  of **(RMP)** is therefore a lower bound on the optimal objective value of  $(\mathbf{TR} - \mathbf{A})$ . We also remark that *Rm-TWCT* does always possess an optimal solution that fulfills the load balancing constraints (3.27). Otherwise, the final job

on machine  $k$  may be transferred to another machine without increasing the total cost (Azizoglu and Kirca, 1999b, Theorem 1). These  $m$  constraints are incorporated into the initial (RMP).

$$\sum_{j=1}^n p_{jk} y_{jk} \leq \frac{1}{m} \left( \sum_j \max_l \{p_{jl}\} + \sum_{l \neq k} \max_j \{p_{jl}\} \right), \quad k = 1, \dots, m. \quad (3.27)$$

The classical textbook application of Benders decomposition iterates between generating Benders cuts based on the optimal solution of the current relaxed master problem and re-optimizing the relaxed master problem with the additional cuts starting from a brand-new search tree. Consequently, the same nodes may be re-visited several times during the course of the Benders decomposition algorithm resulting in an inefficient implementation. With the recent advances in solver technology, a Benders type algorithm may be executed on a single search tree by exploiting the *lazy constraint* feature (IBM ILOG CPLEX, 2012) that allows generating a Benders cut for each candidate incumbent solution. Thus, no integer solution is evaluated more than once, and this generally leads to very substantial computational savings. In-depth discussions are offered in (Rubin, 2011) and at the end of Section 2.4.1 of Chapter 2. The use of the *lazy constraint callback* routine is also reflected in the pseudo-code of our optimal algorithm for (TR – A) stated in Algorithm 3 at the end of this section.

Next, we turn our attention to the efficient generation of the Benders optimality cuts (3.25) by providing a closed form optimal solution to  $(\mathbf{DS}_k - \mathbf{F})$ . Note that  $(\mathbf{DS}_k - \mathbf{F})$  is defined over the entire set of jobs and the full length of the planning horizon  $H$ . However, a job  $j'$  with  $\bar{y}_{j'k} = 0$  is not performed on machine  $k$  and may be excluded from consideration while optimizing  $(\mathbf{DS}_k - \mathbf{F})$ . Therefore, we define  $J_k = \{j \mid \bar{y}_{jk} = 1\}$  as the set of jobs to be processed on machine  $k$  and  $H_k = \sum_{j \in J_k} p_{jk}$  as the associated planning horizon, respectively, and solve a restricted version of  $(\mathbf{DS}_k - \mathbf{F})$  – referred to as  $(\mathbf{DS}_k - \mathbf{R})$  – over these jobs and time periods only. The optimal solution of  $(\mathbf{DS}_k - \mathbf{R})$  is then trivially augmented to an optimal solution of  $(\mathbf{DS}_k - \mathbf{F})$  by setting  $\bar{u}_{jk} = 0$  for  $j \notin J_k$  and  $\bar{v}_{kt} = 0$  for  $t = H_k + 1, \dots, H$ . The validity of this augmentation requires that the objective function coefficients of (TR – A) are non-negative and non-decreasing over time, and that the optimal solution of  $(\mathbf{DS}_k - \mathbf{R})$  satisfies  $\max_{t=1, \dots, H_k} \bar{v}_{kt} = 0$  – see Section 2.4.1 of Chapter 2).

The earlier condition does clearly hold here, and the optimal solution of  $(\mathbf{DS}_k - \mathbf{R})$  specified in (3.28) fulfills the latter because  $\bar{v}_{kt} \leq 0$  for all  $t = 1, \dots, H_k$  and  $\bar{v}_{kH_k} = 0$ . In the presentation below, the jobs assigned to machine  $k$  are re-labeled in the WSPT order; that is,  $i < j$  implies  $\frac{w_i}{p_{ik}} \geq \frac{w_j}{p_{jk}}$  for two jobs  $i, j \in J_k$ . Then,

$$\begin{aligned}\bar{u}_{jk} &= \frac{w_j}{p_{jk}} \left( \sum_{i \leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2} \right) + \sum_{i > j} w_i, & j \in J_k, \\ \bar{v}_{kt} &= \frac{w_{l(t)}}{p_{l(t)k}} \left( t - \sum_{i \leq l(t)} p_{ik} \right) - \sum_{i > l(t)} w_i, & t = 1, \dots, H_k,\end{aligned}\tag{3.28}$$

is an optimal solution for  $(\mathbf{DS}_k - \mathbf{R})$ , where  $l(t)$  is defined such that  $\sum_{i < l(t)} p_{ik} < t \leq \sum_{i \leq l(t)} p_{ik}$ , as we prove next.

**Proposition 3.6.**  $(\bar{u}_k, \bar{v}_k)$  specified in (3.28) is an optimal solution for  $(\mathbf{DS}_k - \mathbf{R})$  with the cost coefficients given in (3.11).

*Proof.* The proof consists of two main steps. First, we show that  $(\bar{u}_k, \bar{v}_k)$  is a feasible solution for  $(\mathbf{DS}_k - \mathbf{R})$ , i.e.,  $\bar{v}_{kt} \leq 0$  for all  $t = 1, \dots, H_k$  and  $\bar{u}_{jk} + \bar{v}_{kt} \leq c_{jkt}$  for all  $j \in J_k$  and  $t = 1, \dots, H_k$ . Then, we demonstrate that the objective function value associated with this feasible solution of the dual slave problem is equal to that of the optimal solution of the corresponding primal problem. Our focus in this proof is entirely on  $(\mathbf{DS}_k - \mathbf{R})$  for a given machine  $k$ , and therefore, every specific job or set of jobs referred to in the following belongs to  $J_k$ .

The non-positivity of  $\bar{v}_{kt}$  for all  $t = 1, \dots, H_k$  follows from

$$\bar{v}_{kt} = \frac{w_{l(t)}}{p_{l(t)k}} \left( t - \sum_{i \leq l(t)} p_{ik} \right) - \sum_{i > l(t)} w_i = -r(t) \frac{w_{l(t)}}{p_{l(t)k}} - \sum_{i > l(t)} w_i \leq 0,$$

where

$$0 \leq r(t) = \sum_{i \leq l(t)} p_{ik} - t \leq p_{l(t)k} - 1,\tag{3.29}$$

and the weights and the processing times are strictly positive.

To show that  $\bar{u}_{jk} + \bar{v}_{kt} \leq c_{jkt}$  is satisfied for all  $j \in J_k$  and  $t = 1, \dots, H_k$ , we substitute the values of  $\bar{u}_{jk}$  and  $\bar{v}_{kt}$  from (3.28) and  $c_{jkt}$  from (3.11). The constraint

$\bar{u}_{jk} + \bar{v}_{kt} \leq c_{jkt}$  then reduces to

$$\begin{aligned}
& \frac{w_j}{p_{jk}} \left( \sum_{i \leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2} \right) + \sum_{i > j} w_i + \frac{w_{l(t)}}{p_{l(t)k}} \left( t - \sum_{i \leq l(t)} p_{ik} \right) - \sum_{i > l(t)} w_i \leq \frac{w_j}{p_{jk}} \left( t + \frac{p_{jk}}{2} - \frac{1}{2} \right) \\
& \iff \frac{w_j}{p_{jk}} \left( \sum_{i \leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2} \right) + \sum_{i > j} w_i - \frac{w_{l(t)}}{p_{l(t)k}} r(t) - \sum_{i > l(t)} w_i \\
& \leq \frac{w_j}{p_{jk}} \left( \sum_{i \leq l(t)} p_{ik} - r(t) + \frac{p_{jk}}{2} - \frac{1}{2} \right) \quad (3.30)
\end{aligned}$$

by replacing  $t - \sum_{i \leq l(t)} p_{ik}$  by  $-r(t)$  and  $t$  by  $\sum_{i \leq l(t)} p_{ik} - r(t)$  based on (3.29). In order to establish the validity of (3.30), we consider the cases  $j \leq l(t)$  and  $j > l(t)$  separately.

If  $j \leq l(t)$ , (3.30) simplifies to

$$\frac{w_j}{p_{jk}} \left( - \sum_{j < i \leq l(t)} p_{ik} \right) + \sum_{j < i \leq l(t)} w_i \leq r(t) \left( \frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}} \right). \quad (3.31)$$

Note that  $j \leq l(t)$  implies  $\frac{w_j}{p_{jk}} \geq \frac{w_{l(t)}}{p_{l(t)k}}$  and leads to  $(p_{l(t)k} - 1) \left( \frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}} \right) \leq r(t) \left( \frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}} \right)$  based on (3.29). Therefore, (3.31) holds if

$$- \sum_{j < i \leq l(t)} p_{ik} \frac{w_j}{p_{jk}} + \sum_{j < i \leq l(t)} p_{ik} \frac{w_i}{p_{ik}} \leq (p_{l(t)k} - 1) \left( \frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}} \right) \quad (3.32)$$

$$\iff \sum_{j < i < l(t)} p_{ik} \left( \frac{w_i}{p_{ik}} - \frac{w_j}{p_{jk}} \right) \leq \frac{w_j}{p_{jk}} - \frac{w_{l(t)}}{p_{l(t)k}} \quad (3.33)$$

is satisfied, where the transition from (3.32) to (3.33) requires adding  $p_{l(t)k} \frac{w_j}{p_{jk}} - w_{l(t)}$  to both sides of (3.32). The inequality (3.33) is clearly correct since  $\left( \frac{w_i}{p_{ik}} - \frac{w_j}{p_{jk}} \right) \leq 0$  for  $i \geq j$  and  $\left( \frac{w_j}{p_{jk}} - \frac{w_{l(t)}}{p_{l(t)k}} \right) \geq 0$ , and this completes the argument for the first case with  $j \leq l(t)$ .

If  $j > l(t)$ , re-arranging the terms of (3.30) leads to

$$\frac{w_j}{p_{jk}} \left( \sum_{l(t) < i \leq j} p_{ik} \right) - \sum_{l(t) < i \leq j} w_i \leq r(t) \left( \frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}} \right). \quad (3.34)$$

The inequality  $\frac{w_j}{p_{jk}} \leq \frac{w_{l(t)}}{p_{l(t)k}}$  follows from  $j > l(t)$ , and we conclude that the right hand side of (3.34) is non-negative. Therefore, in order to prove that (3.34) is satisfied it

is sufficient to demonstrate the correctness of this relation:

$$\sum_{l(t) < i \leq j} p_{ik} \frac{w_j}{p_{jk}} - \sum_{l(t) < i \leq j} p_{ik} \frac{w_i}{p_{ik}} = \sum_{l(t) < i \leq j} p_{ik} \left( \frac{w_j}{p_{jk}} - \frac{w_i}{p_{ik}} \right) \leq 0. \quad (3.35)$$

The validity of inequality (3.35) derives from  $\left( \frac{w_j}{p_{jk}} - \frac{w_i}{p_{ik}} \right) \leq 0$  for  $i \leq j$ . This yields the correctness of (3.30) for the second case with  $j > l(t)$ , and  $(\bar{u}_k, \bar{v}_k)$  is certified as a feasible solution of  $(\mathbf{DS}_k - \mathbf{R})$ .

The primal problem associated with  $(\mathbf{DS}_k - \mathbf{R})$  is the restricted cut generation subproblem  $(\mathbf{TR}_k - \mathbf{R})$  solved over the set of jobs  $J_k$  and the planning horizon  $t = 1, \dots, H_k$ . Based on Corollary 3.5,  $(\mathbf{TR}_k - \mathbf{R})$  is equivalent to the non-preemptive single-machine TWCT problem solved over the jobs in  $J_k$ . Therefore, its optimal objective function value is calculated as  $\sum_{j \in J_k} w_j \left( \sum_{i \leq j} p_{ik} \right)$ , where the completion time of job  $j$  in the non-preemptive WSPT schedule is equal to the sum of the processing times of the jobs placed earlier in the WSPT sequence. Thus, in order to complete the proof, we must argue that the objective function value associated with  $(\bar{u}_k, \bar{v}_k)$  in  $(\mathbf{DS}_k - \mathbf{R})$  is equal to  $\sum_{j \in J_k} w_j \left( \sum_{i \leq j} p_{ik} \right)$ .

$$\begin{aligned} & \sum_{j \in J_k} p_{jk} \bar{u}_{jk} + \sum_{t=1}^{H_k} \bar{v}_{kt} \\ &= \sum_{j \in J_k} p_{jk} \left( \frac{w_j}{p_{jk}} \left( \sum_{i \leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2} \right) + \sum_{i > j} w_i \right) + \sum_{t=1}^{H_k} \left( \frac{w_{l(t)}}{p_{l(t)k}} \left( t - \sum_{i \leq l(t)} p_{ik} \right) - \sum_{i > l(t)} w_i \right) \end{aligned} \quad (3.36)$$

$$= \sum_{j \in J_k} p_{jk} \left( \frac{w_j}{p_{jk}} \left( \sum_{i \leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2} \right) + \sum_{i > j} w_i \right) + \sum_{j \in J_k} \left( \frac{w_j}{p_{jk}} \left( - \sum_{i=0}^{p_{jk}-1} i \right) - p_{jk} \sum_{i > j} w_i \right) \quad (3.37)$$

$$= \sum_{j \in J_k} p_{jk} \left( \frac{w_j}{p_{jk}} \left( \sum_{i \leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2} \right) + \sum_{i > j} w_i \right) - \sum_{j \in J_k} p_{jk} \left( \frac{w_j (p_{jk} - 1)}{2} + \sum_{i > j} w_i \right)$$

$$= \sum_{j \in J_k} p_{jk} \left( \frac{w_j}{p_{jk}} \left( \sum_{i \leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2} - \frac{p_{jk} - 1}{2} \right) \right) = \sum_{j \in J_k} w_j \left( \sum_{i \leq j} p_{ik} \right).$$

For the transition from (3.36) to (3.37), observe that each job  $j \in J_k$  becomes the job  $l(t)$  for  $p_{jk}$  consecutive time periods, and the difference  $\left( t - \sum_{i \leq l(t)} p_{ik} \right)$  runs from  $-(p_{jk} - 1)$  to zero during these time periods. Therefore, we have  $\sum_{t=1}^{H_k} \frac{w_{l(t)}}{p_{l(t)k}} \left( t - \sum_{i \leq l(t)} p_{ik} \right) = \sum_{j \in J_k} \frac{w_j}{p_{jk}} \left( - \sum_{i=0}^{p_{jk}-1} i \right)$ . A similar argument yields  $-\sum_{t=1}^{H_k} \sum_{i > l(t)} w_i = -\sum_{j \in J_k} p_{jk} \sum_{i > j} w_i$ .  $\square$



The general consensus of the literature (Fischetti et al., 2010, Magnanti and Wong, 1981) is that algorithms based on Benders decomposition rarely deliver good computational performance unless the Benders cuts are strengthened. The essence of the matter is to choose a “good” optimal solution of the dual slave problem to generate cuts if primal degeneracy is present in the cut generation subproblem. In the context of this study, the transportation problem is renowned for it is primal degeneracy and an optimal solution of  $(\mathbf{DS}_k - \mathbf{F})$  obtained by extending the optimal solution  $(\bar{\mathbf{u}}_k, \bar{\mathbf{v}}_k)$  of  $(\mathbf{DS}_k - \mathbf{R})$  given in (3.28) by setting  $\bar{u}_{jk} = 0$  for  $j \notin J_k$  and  $\bar{v}_{kt} = 0$  for  $t = H_k + 1, \dots, H$ , results in weak cuts and uncompetitive computational performance. To alleviate this issue, we apply the cut strengthening procedure of Chapter 2 which yields an alternate optimal solution  $(\bar{\mathbf{u}}'_k, \bar{\mathbf{v}}'_k)$  of  $(\mathbf{DS}_k - \mathbf{F})$  – see Proposition 2.3 of Chapter 2:

$$\begin{aligned} \bar{u}'_{jk} &= \bar{u}_{jk}, & j \in J_k, & & \bar{u}'_{jk} &= \min_{t=1, \dots, H_k} (c_{jkt} - \bar{v}_{kt}), & j \notin J_k, \\ \bar{v}'_{kt} &= \bar{v}_{kt}, & t = 1, \dots, H_k, & & \bar{v}'_{kt} &= 0, & t = H_{k+1}, \dots, H. \end{aligned} \quad (3.38)$$

The benefit is that  $y_{jk}$ ,  $j \notin J_k$ , are now added to the right hand side of (3.25) with strictly positive coefficients  $p_{jk}\bar{u}'_{jk}$ ,  $j \notin J_k$ . Note that  $\bar{u}'_{jk} > 0$  for all  $j \notin J_k$  because  $c_{jkt} > 0$  in the entire planning horizon for all jobs and  $\max_{t=1, \dots, H_k} \bar{v}_{kt} = 0$  as discussed just above the presentation of the optimal solution of  $(\mathbf{DS}_k - \mathbf{R})$  in (3.28). The naive calculation of  $\bar{u}'_{jk}$  for all  $j \notin J_k$  requires  $O(nH)$  operations; however, by investigating and exploiting the structure of  $(\bar{\mathbf{u}}_k, \bar{\mathbf{v}}_k)$  we can carry out this calculation in  $O(n)$  time based on Lemma 3.7 and the ensuing discussion. Consequently, the pseudo-polynomial complexity  $O(mnH)$  for strengthening all  $m$  cuts in one iteration of the Benders decomposition algorithm in Chapter 2 is reduced to the polynomial complexity  $O(mn)$  for  $Rm$ -TWCT in this chapter. This enhancement stems from the following result:

**Lemma 3.7.** *For a given job  $j \notin J_k$ , the function  $f_{jk}(t) = c_{jkt} - \bar{v}_{kt}$  defined over  $t = 1, \dots, H_k$  is discrete convex.*

*Proof.* Similar to the convention in the proof of Proposition 3.6, assume that the jobs in  $J_k$  are re-labeled in the WSPT order and define  $l(t)$  such that  $\sum_{i < l(t)} p_{ik} < t \leq \sum_{i \leq l(t)} p_{ik}$ . Obviously,  $l(t)$  is the job processed on machine  $k$  in period  $t$  in the optimal solution of  $(\mathbf{TR}_k - \mathbf{R})$  which schedules all unit jobs of a given job

contiguously by following the WSPT order. Note that  $(t - \sum_{i \leq l(t)} p_{ik}) = -(p_{hk} - 1)$  in the first period  $t$  assigned to a job  $h$ , and this difference is increased by one in each following period job  $h$  is processed until it becomes zero upon the completion of job  $h$ . Consequently, for two consecutive periods  $t, t + 1$  assigned to job  $h$  such that  $l(t + 1) = l(t) = h$ , we have  $\bar{v}_{k,t+1} - \bar{v}_{kt} = \frac{w_h}{p_{hk}}$ . Otherwise, if job  $h$  completes processing in period  $t$  and job  $h + 1$  is started in period  $t + 1$ , then  $l(t) = h, l(t + 1) = h + 1$  and we obtain  $\bar{v}_{k,t+1} - \bar{v}_{kt} = \left(-\frac{w_{h+1}}{p_{h+1,k}} (p_{h+1,k} - 1) - \sum_{i>h+1} w_i\right) - \left(-\sum_{i>h} w_i\right) = \frac{w_{h+1}}{p_{h+1,k}}$ . We conclude that

$$\bar{v}_{k,t+1} - \bar{v}_{kt} = \frac{w_{l(t+1)}}{p_{l(t+1)k}} > 0, \quad t = 1, \dots, H_k - 1.$$

Re-arranging the terms,  $f_{jk}(t) = \left(\frac{w_j}{2} - \frac{w_j}{2p_{jk}}\right) + \left(\frac{w_j}{p_{jk}}t - \bar{v}_{kt}\right)$ , and the difference

$$\begin{aligned} \Delta_{jk}(t) &= f_{jk}(t + 1) - f_{jk}(t) = \left(\frac{w_j}{p_{jk}}(t + 1) - \bar{v}_{k,t+1}\right) - \left(\frac{w_j}{p_{jk}}t - \bar{v}_{kt}\right) \\ &= \frac{w_j}{p_{jk}} - (\bar{v}_{k,t+1} - \bar{v}_{kt}) = \frac{w_j}{p_{jk}} - \frac{w_{l(t+1)}}{p_{l(t+1)k}} \end{aligned} \quad (3.39)$$

is non-decreasing over the interval  $1, \dots, H_k - 1$  because  $\frac{w_j}{p_{jk}}$  is a constant and  $\frac{w_{l(t+1)}}{p_{l(t+1)k}}$  is non-increasing over the interval  $1, \dots, H_k - 1$  based on the WSPT ordering of the jobs in  $J_k$ . This completes the proof because a function  $f : \mathbb{Z}_+ \mapsto \mathbb{R}$  is discrete convex if and only if the differences  $t \mapsto f(t + 1) - f(t)$  are non-decreasing.  $\square$

The discrete convexity of  $f_{jk}(t)$  for  $j \notin J_k$  implies that  $\bar{u}'_{jk} = \min_{t=1, \dots, H_k} (c_{jkt} - \bar{v}_{kt}) = c_{jkt_{jk}^*} - \bar{v}_{kt_{jk}^*}$ , where  $t_{jk}^* = \min\{t = 1, \dots, H_k \mid \Delta_{jk}(t) \geq 0\}$  with the understanding that  $\Delta_{jk}(H_k) \geq 0$ . A further key observation allows us to conduct the search for  $t_{jk}^*$  over the set of jobs in  $J_k$  instead of the set of time periods  $1, \dots, H_k$ . Recall that the optimal solution of  $(\mathbf{TR}_k - \mathbf{R})$  is non-preemptive, and  $l(t) = h$  from period  $\sum_{i<h} p_{ik} + 1$  until period  $\sum_{i \leq h} p_{ik}$ . Consequently, (3.39) assures that  $\Delta_{jk}(t) = \frac{w_j}{p_{jk}} - \frac{w_h}{p_{hk}}$  from period  $C_{h-1} = \sum_{i<h} p_{ik}$  until period  $C_{h-1} + p_h - 1$ , and the period in which  $\Delta_{jk}(t)$  changes sign must coincide with the completion time of a job in  $J_k$  – except when  $\frac{w_j}{p_{jk}} \geq \max_{i \in J_k} \frac{w_i}{p_{ik}}$  and  $t_{jk}^* = 1$ . Moreover, from (3.39) we can also infer that  $t_{j_1k}^* \leq t_{j_2k}^*$  is satisfied for two jobs  $j_1, j_2 \notin J_k$  so that  $\frac{w_{j_1}}{p_{j_1k}} \geq \frac{w_{j_2}}{p_{j_2k}}$ . Putting these ideas together, all  $\bar{u}'_{jk}, j \notin J_k$ , can be computed in  $O(n)$  time by traversing both these jobs and the

jobs in  $J_k$  in the WSPT order – see Algorithm 5.

The pseudo-code of our complete Benders decomposition scheme with the cut strengthening feature for solving (TR – A) is stated in Algorithms 3-5. The finiteness of the algorithm is argued through the finite number of job partitions.

---

**Algorithm 3:** Solving (TR – A) by Benders decomposition and lazy constraint generation.

---

```

1 Create (RMP) with (2.19), (2.20), (2.22), and the load balancing constraints (2.14);
  // Initialization.
2 Invoke CPLEX on (RMP); // Main loop.
3 repeat
4   Identify a new incumbent candidate  $\bar{y}$  with an objective value of  $\sum_{k=1}^m \bar{\eta}_k$ ;
5    $accept\_candidate = true$ ;
6    $[cuts, z_1(\bar{y}), \dots, z_m(\bar{y})] = generate\_cuts(\bar{y})$ ; //  $cuts$  is a collection of  $m$ 
   cuts.
7   for  $k = 1$  to  $m$  do
8     if  $\bar{\eta}_k < z_k(\bar{y})$  then //  $\bar{y}$  violates a missing Benders cut.
9     |   Add  $cuts_k$  to (RMP) as a lazy constraint,  $accept\_candidate = false$ ;
10  until CPLEX determines that the relative optimality gap of the current incumbent is less
   than some threshold;
11 The best available job partition  $\bar{y}^*$  for (TR – A) is retrieved from CPLEX. The
   optimal solution for  $Rm$ -TWCT is obtained by applying the WSPT rule
   independently to the set of jobs on each machine;
```

---



---

**Algorithm 4:** Procedure *generate\_cuts*.

---

```

input : A feasible partition  $\bar{y}$  of the jobs to the machines.
output: Returns  $z_k(\bar{y})$  and a strengthened Benders cut for all machines  $k = 1, \dots, m$ .
1 for  $k = 1$  to  $m$  do
2   Compute the optimal solution  $(\bar{u}_k, \bar{v}_k)$  of  $(DS_k - R)$  as given in (3.28) and
   calculate  $z_k(\bar{y})$ ;
3   Retrieve the job completion times  $C_j, j \in J_k$ , in the associated optimal solution
   of  $(TR_k - R)$ ;
4    $[(\bar{u}'_k, \bar{v}'_k)] = strengthen\_cut(J_k, (C_j, j \in J_k), (\bar{u}_k, \bar{v}_k))$  is an optimal solution of
    $(DS_k - F)$ ;
5   Generate a strengthened Benders cut of the form (2.21) from  $(\bar{u}'_k, \bar{v}'_k)$  and add to
    $cuts$ ;
```

---

---

**Algorithm 5:** Procedure *strengthen\_cut*.

---

**input** :  $J_k$  – Jobs assigned to machine  $k$ , re-labeled in WSPT order,  
 $C_j, j \in J_k$  – Job completion times in the optimal solution of  $(\mathbf{TR}_k - \mathbf{R})$ ,  
 $(\bar{u}_k, \bar{v}_k)$  – Optimal solution of  $(\mathbf{DS}_k - \mathbf{R})$  as specified in (3.28).  
**output**:  $(\bar{u}'_k, \bar{v}'_k)$ .

- 1  $\bar{v}'_{kt} = \bar{v}_{kt}, t = 1, \dots, H_k, \bar{v}'_{kt} = 0, t = H_{k+1}, \dots, H, \bar{u}'_{jk} = \bar{u}_{jk}, j \in J_k$  // no need in actual implementation.
- 2  $i^* = 0, q = \max_{i \in J_k} \frac{w_i}{p_{ik}} = \frac{w_1}{p_{1k}}$  // Job 1 refers to the first job in  $J_k$ .
- 3 **for**  $j \notin J_k$  **do** // Traverse in WSPT order. The entire loop runs in  $O(n)$  time.
- 4     **if**  $q > \frac{w_j}{p_{jk}}$  **then**  $i^* = \max \left\{ i \in J_k \mid i \geq i^*, \frac{w_i}{p_{ik}} > \frac{w_j}{p_{jk}} \right\}$ ;  
      // The search condition  $i \geq i^*$  is justified by  $t_{j_1k}^* \leq t_{j_2k}^*$  for  $j_1, j_2 \notin J_k$   
      with  $\frac{w_{j_1}}{p_{j_1k}} \geq \frac{w_{j_2}}{p_{j_2k}}$  – see (3.39).
- 5     **if**  $i^* = 0$  **then**  $t^* = 1$  **else**  $t^* = C_{i^*}$ ;
- 6      $\bar{u}'_{jk} = c_{jkt^*} - \bar{v}_{kt^*}$ ;

---

### 3.4 Computational Results

The overall goal of our computational study is to demonstrate that the proposed Benders decomposition algorithm – referred to as  $(\mathbf{TR} - \mathbf{A})$ -BDS in the rest of the chapter – has a great computational performance both in absolute and relative terms. We solve instances across a broad range of  $(n, m)$  combinations with both short and long processing times and investigate the effectiveness of our algorithm in order to establish its absolute performance. It turns out that  $(\mathbf{TR} - \mathbf{A})$ -BDS scales very well as instances with up to 1000 jobs and 30 machines are either solved to optimality with a time limit of five minutes or very high-quality incumbents are obtained at termination. For  $m \leq 8$ , the optimal solution is attained within 10 seconds for a great majority of the instances for any  $n$ , and we conclude that  $(\mathbf{TR} - \mathbf{A})$ -BDS is even fast enough to be employed as a subroutine in decomposition algorithms designed for the more general flexible flow- and job shop scheduling problems. Furthermore, to argue that  $(\mathbf{TR} - \mathbf{A})$ -BDS is the best exact algorithm for  $Rm$ -TWCT developed to date, we benchmark it against the CQIP formulation  $(\mathbf{CQ})$  presented in Section 3.3 and solved by an off-the-shelf engine. This approach is referred to as  $(\mathbf{CQ})$ -CPLEX in the sequel. As pointed out in Sections 3.1-3.3,  $(\mathbf{CQ})$ -CPLEX represents the current state-of-the-art for the exact methods designed for  $Rm$ -TWCT. The results reveal that compared to  $(\mathbf{CQ})$ -CPLEX,

(**TR – A**)-**BDS** either determines the optimal solution in considerably shorter time or it identifies an incumbent of substantially higher quality at the time limit. The details of our analyses are presented in the following.

To facilitate a direct comparison, our instance generation follows suit with that of [Plateau and Rios-Solis \(2010\)](#) who evaluated (**CQ**) empirically. For each job  $j \in \{1, \dots, n\}$ , the processing time  $p_{jk}$  on machine  $k \in \{1, \dots, m\}$  and the unit completion time penalty  $w_j$  are drawn from the discrete uniform distribution  $U[1, 20]$ . We create 10 instances for each combination of  $n \in \{30, 100, 400, 1000\}$  and  $m \in \{2, 4, 6, 8, 16, 30\}$ , except for  $n = 30$  and  $m = 16, 30$ , where the average number of jobs per machine is too few. In this setup, the ratio  $\frac{n}{m}$  varies between 3.33 and 500 which allows us to explore the sensitivity of (**TR – A**)-**BDS** to this parameter. Note that the branch-and-price algorithms of [Chen and Powell \(1999b\)](#) and [van den Akker et al. \(1999\)](#) mentioned in Section 3.1 run into trouble for  $\frac{n}{m} > 10$ . Furthermore, recall that the size of (**TR – A**) is pseudo-polynomial and depends on the length of the processing times. Therefore, in an effort to verify the robustness of (**TR – A**)-**BDS** with respect to the range of the processing times, we repeat the same generation scheme except that the processing times are drawn from the discrete uniform distribution  $U[1, 100]$  – i.e.,  $p_{\max} = 100$  – which brings the total number of instances solved in this study to 440.

The computational results are obtained on a personal computer with a 2.33 GHz Intel® Core™2 Quad processor Q8200 and 8 GB of memory running on Windows 7. (**TR – A**)-**BDS** is implemented in C++ using the Concert Technology component library of IBM® ILOG® CPLEX® 12.5. Under the default parameter settings, the implementation of a control callback – such as the lazy constraint callback – leads CPLEX to turn off its dynamic search feature and apply a traditional branch-and-cut strategy with a single thread ([IBM ILOG CPLEX, 2012](#)). Therefore, to exploit parallelism and promote simultaneous cut generation, CPLEX is allowed to use up to four parallel threads – as specified by the Threads parameter – with the ParallelMode switch set to Opportunistic. Moreover, based on the positive previous experience in Chapter 2 the MIPEmphasis switch, which “controls the trade-offs between speed, feasibility, optimality, and moving bounds in MIP,” takes on the value four in order to emphasize finding high-quality hidden feasible solutions. (**CQ**)-**Cplex** calls CPLEX to solve (**CQ**) with the default parameter set-

tings, except that `Threads=4`, `ParallelMode=Opportunistic`, and `MIPEmphasis=4` for a fair comparison with **(TR – A)-BDS**. In both methods, **CPLEX** terminates the optimization if the relative optimality gap drops below  $\text{EpGap}=10^{-3}=0.1\%$ , or the working memory exceeds `WorkMem=5120=5 GB`, or the time expended reaches `TiLim=300` seconds. More details on these parameters are available in ([IBM ILOG CPLEX, 2012](#)).

Table 3.1 consists of 22 rows, one for each possible combination of  $n$  and  $m$  listed in the first two columns. Each figure in the table represents a statistic over 10 instances. The number of instances solved to optimality within the time limit appears in the columns labeled with “#”, and the columns under “%Gap” and “Time” present the average optimality gaps retrieved from **CPLEX** at termination and the average solution times, respectively. Note that **CPLEX** uses the formula  $\frac{|best\_bound - best\_integer|}{10^{-10} + |best\_integer|}$  for computing the optimality gap of an instance ([IBM ILOG CPLEX, 2012](#)), where *best\_bound* is the largest available lower bound and *best\_integer* is the objective value of the incumbent at termination. A color formatting scheme is applied separately to each of the three performance measures “%Gap”, “Time”, and “#,” so that the values ranging from better to worse are indicated with colors changing from green towards red. The results for instances with relatively short processing times are reported in the left half of the table in Columns 3-8 under the heading “ $p_{\max} = 20$ .” The remaining columns depict the performance measures for the corresponding instances with  $p_{\max} = 100$ .

The results in Table 3.1 underline that **(TR – A)-BDS** provides provably optimal solutions for the majority of the instances well within the time limit of 300 seconds. More specifically, **(TR – A)-BDS** solves 343 out of a total of 440 instances to optimality in 3.73 seconds on average with a maximum solution time of 162.48 seconds. In contrast, **(CQ)-CPLEX** attains only 270 optimal solutions in 13.95 seconds on average with a maximum of 241.51 seconds. The average and maximum gaps of **(TR – A)-BDS** for those 97 instances that could not be solved to optimality within the specified time limit are just 0.96% and 5.51%, respectively. The corresponding figures for **(CQ)-CPLEX** are 5.21% and 75.45% over 150 instances. **(CQ)-CPLEX** chokes on the remaining 20 largest instances with  $n = 1000$ ,  $m = 30$  and terminates due to an out-of-memory error. The differences between **(TR – A)-BDS** and **(CQ)-CPLEX** become more apparent if we separate

**Table 3.1** Average optimality gap and solution time results for  $Rm$ -TWCT.

		$p_{\max} = 20$						$p_{\max} = 100$					
		(TR – A)-BDS			(CQ)-CPLEX			(TR – A)-BDS			(CQ)-CPLEX		
$n$	$m$	%Gap	Time	#	%Gap	Time	#	%Gap	Time	#	%Gap	Time	#
30	2	0.07	0.06	10	0.03	0.05	10	0.04	0.06	10	0.04	0.04	10
	4	0.03	0.11	10	0.04	0.15	10	0.02	0.14	10	0.02	0.14	10
	6	0.01	0.22	10	0.05	0.54	10	0.01	0.30	10	0.07	0.71	10
	8	0.06	0.60	10	0.09	3.91	10	0.05	0.67	10	0.06	1.06	10
100	2	0.08	0.13	10	0.08	0.14	10	0.07	0.13	10	0.05	0.11	10
	4	0.08	1.00	10	0.09	1.60	10	0.08	1.07	10	0.10	1.78	10
	6	0.09	1.62	10	0.26	70.08	8	0.08	1.61	10	0.40	123.54	6
	8	0.10	15.21	10	0.71	153.28	5	0.09	9.15	10	1.13	219.34	3
	16	0.67	273.19	1	5.91	182.04	4	0.84	260.67	2	7.30	300.01	0
	30	2.12	300.01	0	16.13	300.02	0	3.80	300.01	0	30.49	300.02	0
400	2	0.06	0.05	10	0.03	1.16	10	0.05	0.05	10	0.01	1.47	10
	4	0.05	0.56	10	0.07	3.28	10	0.06	0.41	10	0.07	5.79	10
	6	0.07	0.82	10	0.22	240.67	2	0.07	0.70	10	0.16	191.02	4
	8	0.09	1.03	10	0.30	151.50	5	0.09	1.72	10	0.28	239.19	3
	16	0.13	300.02	0	1.86	248.95	3	0.22	300.01	0	1.86	300.09	0
	30	0.39	300.03	0	3.75	300.17	0	0.66	300.02	0	6.00	300.15	0
1000	2	0.05	0.10	10	0.01	13.22	10	0.05	0.10	10	0.00	19.21	10
	4	0.04	0.87	10	0.03	27.43	10	0.06	0.79	10	0.02	34.66	10
	6	0.04	1.72	10	0.09	109.50	8	0.04	1.46	10	0.04	49.49	10
	8	0.05	2.67	10	0.14	247.60	2	0.04	1.97	10	0.11	203.59	4
	16	0.09	51.00	10	0.83	257.90	2	0.09	5.93	10	0.63	281.08	1
	30	0.30	300.11	0	-	-	-	0.17	300.17	0	-	-	-

out the groups of instances solved to optimality by both methods and those not solved to optimality by either method within the time limit. On 198 of the 263 instances in the earlier group, (TR – A)-BDS outpaces (CQ)-CPLEX by an average (& maximum) factor of 32.48 (& 228.63) computed from the ratios of the solution times of (CQ)-CPLEX to those of (TR – A)-BDS. On six instances the solution times are identical, and on the remaining 59 instances (CQ)-CPLEX is on average 2.44 times faster, where the corresponding maximum is 10.90. In the second group of 70 instances, (TR – A)-BDS attains a smaller optimality gap at termination for 67 instances. The difference in the optimality gaps is on average 9.37% and reaches a maximum of 70.93%. (CQ)-CPLEX yields a smaller terminal gap on just three instances and the difference does not exceed 1.81%. In addition, note that there are only 7 instances for which (TR – A)-BDS is only able to provide an incumbent at the time limit while (CQ)-CPLEX solves these instances optimally. In comparison,

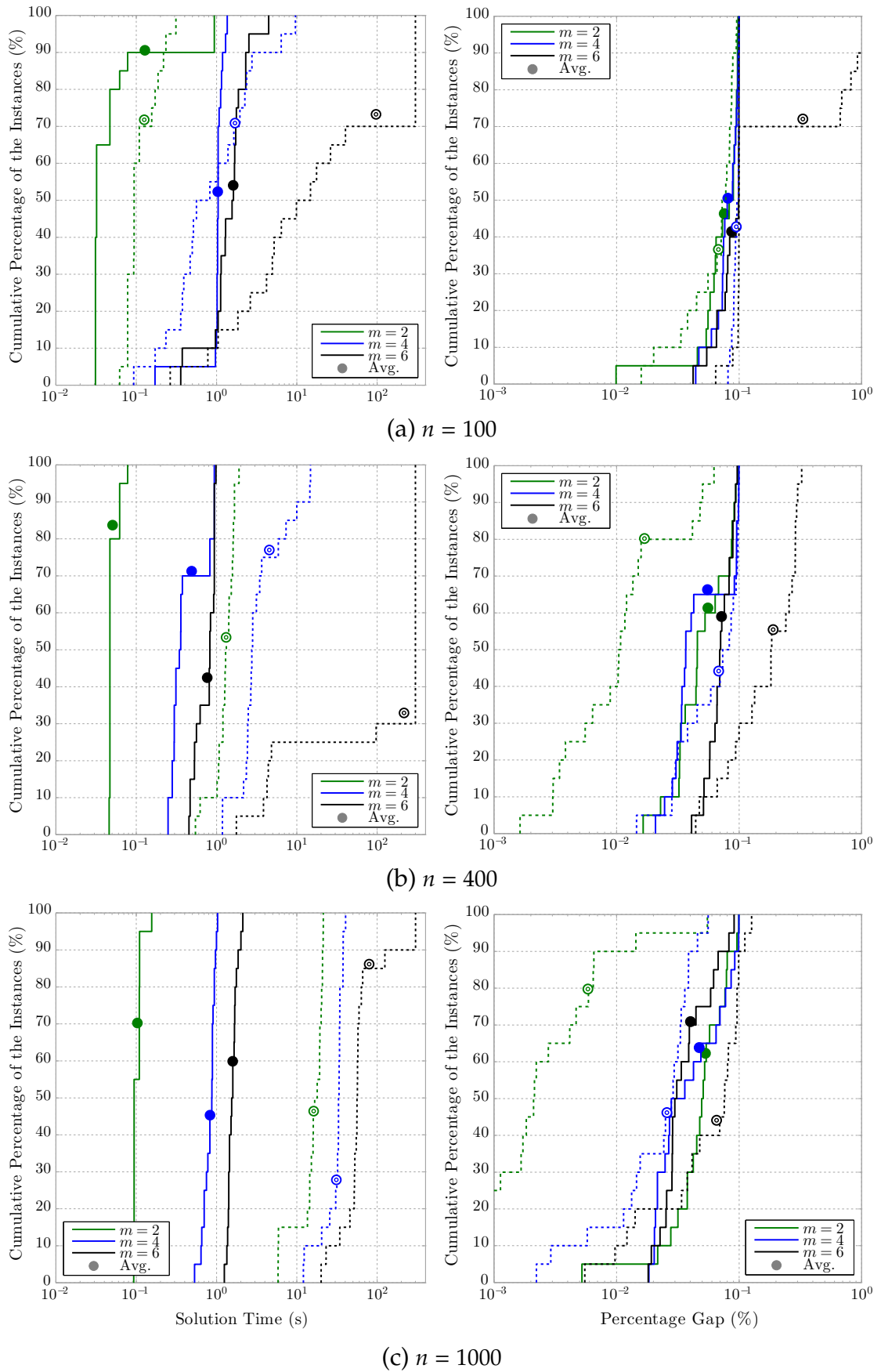
(**TR – A**)-**BDS** supplies optimal solutions for 80 instances that remain unsolved at the time limit by (**CQ**)-**CPLEX** and obtains incumbents very close to optimality with an average gap of 0.24% for the 20 instances with  $n = 1000$ ,  $m = 30$ , while these instances are completely beyond the reach of (**CQ**)-**CPLEX** due to insufficient memory. To conclude, we stress that (**TR – A**)-**BDS** is clearly the exact algorithm of choice for *Rm-TWCT* because it either delivers an optimal solution substantially faster or provides an incumbent with a much smaller optimality gap at termination.

Table 3.1 attests to the solid performance of (**TR – A**)-**BDS** regardless of the range of the processing times. The performance indicators related to (**TR – A**)-**BDS** for both  $p_{\max} = 20$  and  $p_{\max} = 100$  are similar. We reckon that two factors are at play here. First, the magnitude of the processing times has no effect on the size of (**RMP**) and the number of job-to-machine assignments, and the pseudo-polynomial size of (**TR – A**) is therefore completely relegated to the dual slave problems. Second, the analytic solution of ( $\mathbf{DS}_k - \mathbf{F}$ ) offsets the pseudo-polynomial size issue in practice.

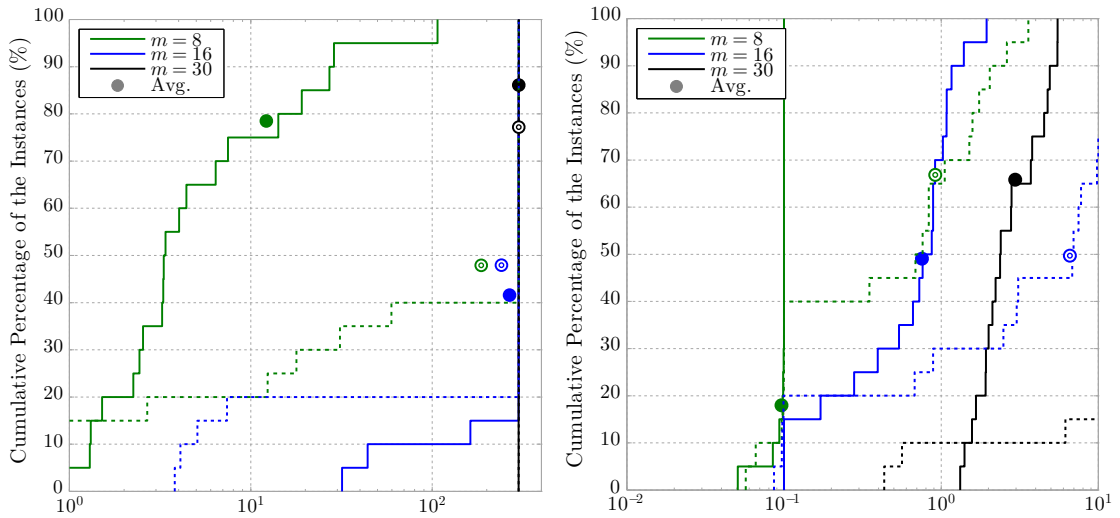
Next, we investigate how (**TR – A**)-**BDS** and (**CQ**)-**CPLEX** scale with the number of jobs and machines. For a fixed  $n$ , the solution times of (**TR – A**)-**BDS** and (**CQ**)-**CPLEX** increase with  $m$ . That is, both methods favor larger  $\frac{n}{m}$  ratios. This may be regarded as a significant advantage over the branch-and-price algorithms of [Chen and Powell \(1999b\)](#) and [van den Akker et al. \(1999\)](#) which perform better for  $\frac{n}{m} \leq 10$ . Clearly, the more likely practical scenario is that  $n$  is significantly larger than  $m$ . Furthermore, observe that the solution times of (**TR – A**)-**BDS** do not necessarily degrade with increasing  $n$  for a fixed  $m$ . Loosely speaking, the computational performance of (**TR – A**)-**BDS** is determined by the number of machines. In contrast, the performance of (**CQ**)-**CPLEX** suffers from both higher  $n$  and  $m$  values.

Figures 3.1-3.2 further substantiate the robustness and scalability of (**TR – A**)-**BDS** as an exact approach for *Rm-TWCT*. The empirical distributions of the solution times and the optimality gaps associated with both methods are depicted in these figures, where each curve is based on 20 instances. The horizontal axes are in logarithmic scale to increase the readability of the graph. The median solution times and optimality gaps are associated with the 50% mark on the vertical axis,

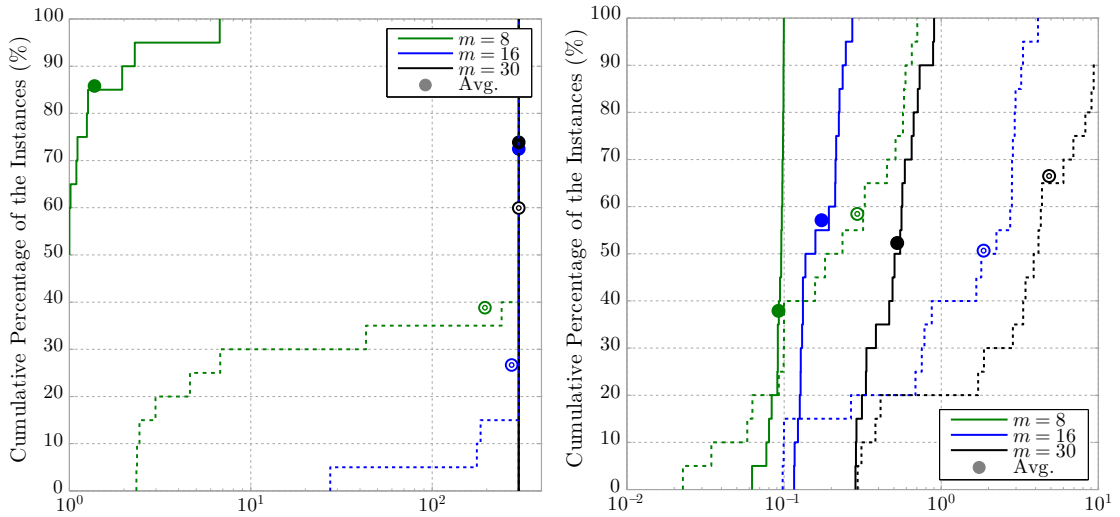




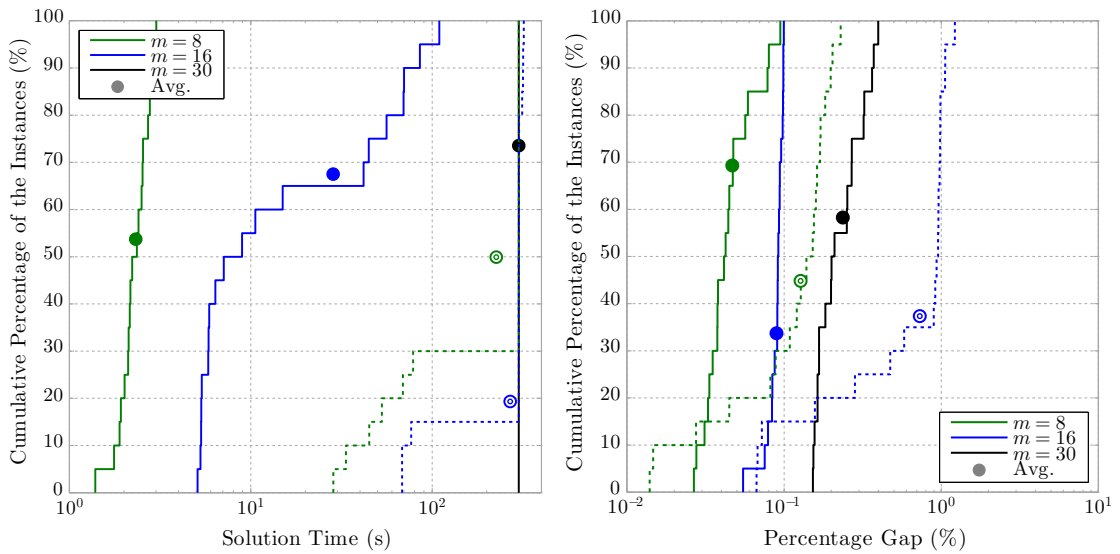
**Figure 3.1** The empirical distributions of the solution times and the optimality gaps of (TR – A)-BDS (—) and (CQ)-CPLEX (– –) for  $Rm$ -TWCT instances with 2, 4, and 6 machines.



(a)  $n = 100$



(b)  $n = 400$



(c)  $n = 1000$

**Figure 3.2** The empirical distributions of the solution times and the optimality gaps of (TR – A)-BDS (—) and (CQ)-CPLEX (– –) for  $Rm$ -TWCT instances with 8, 16, and 30 machines.

and the average gaps are explicitly indicated. Note that the shape of the optimality gap curves to the left of the  $10^{-1}\%$  mark do not bear any meaning because the relative optimality gap parameter of CPLEX is set to  $\text{EpGap} = 10^{-3} = 10^{-1}\%$ . The relative insensitivity of **(TR – A)-BDS** to  $n$  for a fixed  $m$  is also evident from Figure 3.1, where the curves for a fixed  $m$  are stacked on top of each other from Figure 3.1a toward Figure 3.1c. As stated previously, the number of machines is the main determinant of the solution time of **(TR – A)-BDS**; the curves for a fixed  $n$  shift from left to right as  $m$  increases. Furthermore, we can also claim that **(TR – A)-BDS** demonstrates a very consistent performance for these instances because the solution time curve for a given  $(n, m)$  combination rises sharply and exhibits little variability across instances. Figure 3.1 confirms that the solution time performance of **(TR – A)-BDS** is superior to that of **(CQ)-CPLEX** because the curves for **(TR – A)-BDS** generally lie to the left of the corresponding curves for **(CQ)-CPLEX**. Figure 3.2 is less informative with respect to the solution times because both methods often hit the time limit for these instances. However, the optimality gap curves of **(TR – A)-BDS** clearly dominate those of **(CQ)-CPLEX**. Overall, we may draw the conclusion that **(TR – A)-BDS** is a scalable exact algorithm for  $Rm\text{-}TWCT$  and does either find the optimal solution faster than the current state-of-the-art in the literature or it identifies better incumbents at termination.

## CHAPTER 4

# LOGIC-BASED BENDERS DECOMPOSITION FOR COMMON DUE DATE TOTAL WEIGHTED EARLINESS/TARDINESS

In this study, we develop a computationally effective logic-based Benders decomposition (LBBD) algorithm for the unrelated parallel machine unrestrictive common due date total weighted earliness/tardiness (UCDD) scheduling problem. The key contributions of this chapter are twofold. On the one hand, we offer a viable solution approach for solving this strongly  $\mathcal{NP}$ -hard scheduling problem. The computational results indicate that the proposed solution approach is clearly the exact algorithm of choice for this problem because it either delivers an optimal solution substantially faster than the state-of-the-art algorithm or provides an incumbent with a much smaller optimality gap at termination. Furthermore, the proposed algorithm thrives on the instances with large number of jobs as instances with 1000 jobs and up to 6 machines are solved to optimality within just 17 seconds. On the other hand, we demonstrate that by studying the combinatorial structure of the problem, it is possible to devise a scalable LBBD algorithm for a scheduling problem with an irregular minsum objective function – i.e., UCDD. This aspect is missing in the existing LBBD literature as mainly scheduling problems with regular performance measures are tackled and the results for problems

with minsum objectives are not on a par with those for minmax objectives.

## 4.1 Introduction

In this chapter, we address a fundamental scheduling problem of minimizing the total weighted earliness/tardiness with respect to an unrestrictive common due date in the unrelated parallel machine environment. The performance measure of the problem has been the subject of many studies within the last three decades, as it captures the scheduling aspect of the just-in-time philosophy. That is, a job should be completed only when it is required. This ensures that the costs associated with the jobs that are completed before the due date – e.g., insurance, storage and perishing costs – and the costs due to the contractual liabilities and the loss of customer goodwill are both minimized simultaneously. Formally, we characterize the problem we consider as  $Rm/d_j = d^l / \sum_j \epsilon_j E_j + \pi_j T_j$  ( $Rm$ -UCDD) following the three field notation of [Graham et al. \(1979\)](#) in classifying scheduling problems. The notation  $Rm$  in the first field stands for a bank of  $m$  unrelated machines where  $d_j = d^l$  stands for an unrestrictively large common due date. The earliness and tardiness of job  $j$  are represented by  $E_j$  and  $T_j$ , respectively, and  $\epsilon_j$  and  $\pi_j$  are the associated unit weights.  $Rm$ -UCDD is strongly  $\mathcal{NP}$ -hard because of the strongly  $\mathcal{NP}$ -hard identical parallel machine scheduling problem  $Pm/d_j = d^l / \sum_j w_j (T_j + E_j)$  ([Webster, 1997](#)).

The review of the related parallel machine scheduling literature in Section [4.2.1](#) reveals the absence of a scalable algorithm for  $Rm$ -UCDD. More specifically, to the best of our knowledge, the only solution approach for  $Rm$ -UCDD is to solve a monolithic convex quadratic integer programming (CQIP) formulation and this method is effective only for small instances ([Plateau and Rios-Solis, 2010](#)). We benchmark the performance of our LBB algorithm against that of this formulation in Section [4.4](#).

Motivated by the considerations outlined above and in Chapter [1](#), our main goal in this chapter is to develop an efficient and effective exact solution approach for solving  $Rm$ -UCDD. To this end, we make use of the LBB framework of [Hooker and Ottosson \(2003\)](#) which is proven to be useful solving planing and scheduling problems with regular performance measures. One of the main contributions of

this chapter is to offer a computationally effective LBB algorithm for a scheduling problem with a non-regular minsum objective function. Note that even though LBB algorithms have been successfully utilized for regular objective functions, it is known that irregular performance measures lead to new methodological issues in the design of solution approaches (Baker and Scudder, 1990, Kanet and Sridharan, 2000). Moreover, the review in Section 4.2.2 shows that existing LBB algorithms work significantly better for minmax scheduling objectives such as makespan and for other basic objectives such as finding a feasible solution and minimizing job-to-machine assignment costs. However, when the objective is a minsum objective such as minimizing total tardiness, the performance of the algorithms quickly deteriorates due to the fact that it is difficult to compute strong lower bounds for additive scheduling objective functions.

Another contribution of this chapter is that our algorithm is by far the best performing exact algorithm up to date for solving *Rm-UCDD*. The computational results in Section 4.4 illustrate that our exact method solves all instances with 2 machines and up to 1000 jobs to optimality within 3.4 seconds. Even though the instances with 4 and 6 machines are more time consuming, all except one and almost half of the instances with 4 and 6 machines are solved to optimality with average solution times of 123 and 393 seconds, respectively. Furthermore, the optimality gaps of the instances with 4 and 6 machines that are not solved within the time limit of 1 hour are almost always less than 3% with a maximum (& average) of 5.66% (& 1.47%).

The remainder of the chapter consists of four sections. In the next section, we review the related literature to position our work. In Section 4.3, we first summarize the theory of logic-based Benders decomposition and present our LBB algorithm for *Rm-UCDD*. This is followed in Section 4.4 by the computational experiments which demonstrate the efficacy of our approach.

## 4.2 Review of Related Literature

This review is constructed in two sections such that we position our work with respect to the literature on unrelated parallel machine scheduling and LBB separately. This enables us to underline the contributions of this work in both

research areas. Note that the performance figures presented in this section are obtained by their respective authors on different computing platforms.

### 4.2.1 Parallel Machine Scheduling

The common due date scheduling problem was first introduced more than 3 decades ago (Kanet, 1981) and it has been studied from different angles; however, there are only a handful studies on the unrelated parallel machine environment. Comprehensive reviews of the early work on the common due date scheduling problems are given by Baker and Scudder (1990) and the reader is referred to the survey paper by Lauff and Werner (2004) and the literature review in (Rios-Solis and Sourd, 2008) for further information and additional references on the common due date problems. A detailed discussion of the parallel machine scheduling literature on additive due date related performance measures is presented in Section 2.2 of Chapter 2. In this review, we restrict our attention to the literature related to the parallel machine common due date earliness/tardiness scheduling problems since this part of the literature creates the context for our study and provide a few important pointers otherwise.

The computational results presented in Section 2.5 of Chapter 2 and in Section 3.4 of Chapter 3 clearly demonstrate the effectiveness of the mathematical programming based decomposition approaches for the unrelated parallel machine scheduling problems. Other examples include (Chen and Powell, 1999a), in which the authors consider a special case of *Rm-UCDD*, in which all machines are identical and obtain a set partitioning model of the problem through Dantzig-Wolfe reformulation. The linear programming (LP) relaxation of the set partitioning reformulation yields tight lower bounds, and instances with up to 60 jobs and 6 machines are solved to optimality. In a related study, Chen and Lee (2002) extend this approach by incorporating a common due date window and instances with up to 40 jobs and any number of machines are solved to optimality within reasonable times. Rios-Solis and Sourd (2008) consider the same problem as Chen and Powell (1999a), except that they allow for the common due date to be restrictively small. The main contribution of this work is a pseudo-polynomial time dynamic programming algorithm that can identify the best schedule in an exponential-size

neighborhood of the current solution. [Soukhal and Toung \(2012\)](#) study the special cases of several single- and uniform parallel machine (un-)restricted common due date total (un-)weighted earliness/tardiness scheduling problems in which all processing times are equal. They present dominance properties and polynomial and exponential time algorithms for these problems.

Finally, [Plateau and Rios-Solis \(2010\)](#) is the first study available on common due date problems in the unrelated parallel machine environment that designs an optimal algorithm. Inspired by the CQIP formulation of [Skutella \(2001\)](#) – who proposes this formulation as a means of developing an approximation algorithm for total weighted completion time (TWCT) with a performance guarantee of  $3/2$  – the authors first perform an experimental study on this formulation. Then, based on the success of the results, they develop CQIP reformulations to solve both *Rm-UCDD* and  $Rm/d_j = d^r / \sum_j \epsilon_j E_j + \pi_j T_j$  (*Rm-RCDD*) where  $d_j = d^r$  stands for a restrictive common due date. For the first problem, the authors apply the Diagonal Perturbation Method (DPM) for convexification and instances with up to 4 machines and 50 jobs are solved optimally within one hour. We present this formulation in Section 4.3 and benchmark the performance of our LBBD algorithm against it in Section 4.4.

For *Rm-RCDD*, the DPM procedure, however, cannot be adapted and the authors have to resort to a different procedure: the QCR method developed by [Billionnet et al. \(2009\)](#). This method perturbs all the elements of the Hessian matrix by solving a semidefinite relaxation of the problem to convexify the objective function and obtain a tighter continuous lower bound. The results are not satisfactory because the method is too time consuming and instances with 30 jobs and 2 machines are not solved within the time limit of 2 hours. Later, [Beyranvand et al. \(2012\)](#) show that the formulation of [Plateau and Rios-Solis \(2010\)](#) does not describe the true feasible region of *Rm-RCDD*. By adding some constraints, they slightly change the feasible region while ensuring that the results of [Plateau and Rios-Solis \(2010\)](#) remain correct for this modified model. They also describe the incorrect use of the QCR method by [Plateau and Rios-Solis \(2010\)](#) and specialize it for the new model. They do not report CPU times and they are only able to solve instances with up to 10 jobs and 2 machines.



### 4.2.2 LBB in Scheduling

The general theory of logic-based Benders decomposition is presented in (Hooker and Ottosson, 2003). The core idea of LBB is the same as that of Benders decomposition which is “to learn from mistakes”, but LBB extends this notion to a larger class of problems. The key difference of LBB from the classical Benders decomposition approaches is that it does not derive the Benders cuts from the LP dual of the subproblem (Benders, 1962), but makes use of the *inference duality* instead. The inference dual of a problem is the problem of inferring the tightest bound from the constraints of the primal problem and a solution to this dual problem takes the form of a logical deduction. This logical deduction provides a valid bound for the subproblem and yields a Benders cut. Therefore, in theory, logic-based Benders cuts can be obtained from any form of subproblem; however, they must be tailored for each class of problems individually. This, in turn, paves the way for exploiting the problem structure and combining mixed integer linear programming (MIP) and constraint programming (CP) (Hooker, 2007a). Noting that CP methods are well suited for solving scheduling problems, Hooker (2000) suggests this framework for solving planning and scheduling problems.

The first work which makes use of this scheme for solving a machine scheduling problem is due to Jain and Grossmann (2001). Motivated by the work of Bockmayr and Kasper (1998), the authors develop algorithms, which use two incomplete models – i.e., a relaxed MIP model and a CP feasibility model – that are mutually complementary, to solve a class of unrelated parallel machine scheduling problems in which only a subset of binary variables have non-zero coefficients in the objective function. That is, only the problems with a fixed assignment cost based objective are within the scope of their method. They demonstrate the effectiveness of their MIP/CP method on an unrelated parallel machine scheduling problem in which there is a cost of processing a job on a machine. They report two to three orders of magnitude speed improvement over the standalone MIP and CP models.

Later, Thorsteinsson (2001) proposes an approach which closely resembles the LP/CP based branch-and-bound method outlined in (Jain and Grossmann, 2001) and solves the same problem with the MIP/CP based decomposition method pro-

posed in (Jain and Grossmann, 2001), except that the author does not solve the master problem to optimality in each iteration and achieves substantial computational savings. Moreover, Thorsteinsson notes that the success of this method is due to the additional valid inequalities included in the master problem and without them the solution time increases substantially.

Bockmayr and Pizaruk (2003) and Sadykov and Wolsey (2006) present several versions of this hybrid approach for the same unrelated parallel machine scheduling problem. The former tests several heuristics to generate extra Benders cuts within their hybrid MIP/CP branch-and-cut algorithm. Sadykov and Wolsey (2006) strengthen the master problem with valid inequalities and test a total of seven different hybrid and pure MIP, MIP/CP, and column generation (CG) algorithms. They find that the performances of two hybrid algorithms, MIP+/CP and CG-MIP+/CP, dominate the other approaches. They also note that the tightness of the MIP formulation plays an important role in the convergence of the algorithms.

Similar hybrid strategies have been used in decomposition approaches for solving various planning and scheduling problems in different fields. These include steel production scheduling (Harjunkoski and Grossmann, 2001), multistage batch scheduling (Harjunkoski and Grossmann, 2002), production planning in a chemical plant (Maravelias and Grossmann, 2004, Timpe, 2002), multi-processor scheduling (Cambazard et al., 2004), allocation and scheduling of multi-processor systems-on-chips (Benini et al., 2005), double round robin tournament scheduling (Rasmussen and Trick, 2007), integrated shift-selection and task-sequencing (Barlatt et al., 2010), and multiple resource cumulative scheduling (Ciré et al., 2015).

The common denominator of all these applications is that the objective is either just finding a feasible solution or it is a function of only the master problem variables. That is, the subproblems do not take part in the optimization and they are only for ensuring the feasibility. This considerably simplifies the process of creating valid Benders cuts. Nevertheless, there exists several LBB applications which explicitly use optimality cuts – i.e., the subproblem is not only a feasibility problem, but its objective function value contributes to that of the original problem. Hooker (2004, 2005a,b, 2006, 2007b) uses LBB for solving several planning and scheduling problems, in which the objectives are minimizing makespan, the

number of late jobs, or the total tardiness. The author notes that the performance of the algorithm for minimizing makespan is on par with that when the subproblem is just a feasibility problem. However, the performance quickly deteriorates when the objective is of minsum type, instead of minmax. [Coban and Hooker \(2013\)](#) adapt LBBDD to a segmented single-machine scheduling problem in which each job must be completely processed within one segment of the time horizon. The problem naturally decomposes by the segments, and the actual scheduling of the jobs are handled by the subproblems. The authors consider three objectives – i.e., finding a feasible solution, minimizing makespan, and minimizing total tardiness –, and note that the proposed method scales up much more effectively on the instances of the feasibility and makespan problems.

Even though logic-based Benders cuts have been developed for optimization subproblems, the objective function is always a regular performance measure in all of these applications. Note that it is well established that designing a solution approach for a non-regular performance measures has its own peculiarities ([Baker and Scudder, 1990](#), [Kanet and Sridharan, 2000](#)). Furthermore, the existing algorithms do not perform as well for minsum objective functions as they do for minmax objectives. The reason is the lack of strong lower bounds for additive scheduling objectives which translates into weak optimality cuts ([Şen and Bülbül, 2015b](#)). Nevertheless, in this work, we demonstrate that very effective Benders cuts can be created for a problem with a non-regular minsum objective function by studying the combinatorial structure of the problem.

### 4.3 Solution Approach

In the *Rm-UCDD* problem, there are  $n$  jobs and  $m$  unrelated parallel machines, which are all ready at time zero. A machine can execute at most one job at a time and each job is required to receive a non-preemptive service from exactly one of the machines, where the processing of job  $j$  on machine  $k$  takes an integer duration of  $p_{jk}$  time units. An unrestrictive common due date  $d$  – also assumed to be integral – is associated with each job  $j$ . If job  $j$  completes processing before (or after)  $d$ , a penalty  $\epsilon_j$  (or  $\pi_j$ ) per unit time is incurred. Thus, as introduced in Section 4.1, the total weighted earliness/tardiness over all jobs is determined as  $\sum_j \epsilon_j E_j + \pi_j T_j$

where the earliness and tardiness of job  $j$  are calculated as  $E_j = \max(0, d - C_j)$  and  $T_j = \max(0, C_j - d)$ , respectively, and  $C_j$  denotes the completion time of job  $j$ .

As mentioned in Sections 4.1 and 4.2.1, the CQIP formulation presented in (Plateau and Rios-Solis, 2010) exhibits the best computational performance on  $Rm$ -UCDD to date and is stated below. We benchmark against this formulation in Section 4.4.

$$\begin{aligned}
(\text{CQ} - \text{U}) \quad & \text{minimize} \quad \sum_{j=1}^n \sum_{k=1}^m \left( \frac{1}{2} \epsilon_j p_{jk} \left( (y_{kj}^E)^2 - y_{kj}^E \right) + \sum_{i <_k^E j} \epsilon_j p_{ik} y_{ki}^E y_{kj}^E \right) \\
& + \sum_{j=1}^n \sum_{k=1}^m \left( \frac{1}{2} \pi_j p_{jk} \left( (y_{kj}^T)^2 + y_{kj}^T \right) + \sum_{i <_k^T j} \pi_j p_{ik} y_{ki}^T y_{kj}^T \right) \quad (4.1)
\end{aligned}$$

$$\text{subject to} \quad \sum_{k=1}^m y_{kj}^E + y_{kj}^T = 1, \quad j = 1, \dots, n, \quad (4.2)$$

$$y_{kj}^E, y_{kj}^T \in \{0, 1\}, \quad j = 1, \dots, n, \quad k = 1, \dots, m. \quad (4.3)$$

In (CQ – U), the binary variable  $y_{kj}^E$  (&  $y_{kj}^T$ ) takes the value 1 if job  $j = 1, \dots, n$ , is assigned to the early (& tardy) side of machine  $k = 1, \dots, m$ , and is zero otherwise. The notation  $i <_k^E j$  indicates that either  $\frac{\epsilon_i}{p_{ik}} > \frac{\epsilon_j}{p_{jk}}$  or  $\frac{\epsilon_i}{p_{ik}} = \frac{\epsilon_j}{p_{jk}}$  and  $i < j$  – i.e., the completion time of job  $i$  should be closer to the common due date  $d$  than that of job  $j$ , if both jobs are scheduled early on machine  $k$ . Similarly,  $i <_k^T j$  implies that either  $\frac{\pi_i}{p_{ik}} > \frac{\pi_j}{p_{jk}}$  or  $\frac{\pi_i}{p_{ik}} = \frac{\pi_j}{p_{jk}}$  and  $i < j$ . (CQ – U) relies on the basic observation that the completion time of job  $j$  is either  $d - \sum_{k=1}^m y_{kj}^E \left( \sum_{i <_k^E j} p_{ik} y_{ki}^E \right)$  or  $d + \sum_{k=1}^m y_{kj}^T \left( p_{jk} + \sum_{i <_k^T j} p_{ik} y_{ki}^T \right)$ , and on the convexification of the resulting objective.

### 4.3.1 Overview of LBB

The logic-based Benders decomposition is a generalization of the classical Benders decomposition. As it is the case for Benders decomposition, LBB uses a strategy of *learning from mistakes* (Hooker and Ottosson, 2003); however, the key difference is how the consequences of these “mistakes” are extracted and employed in the solution process.

Similar to the Benders decomposition, LBB partitions the variables of a problem into two vectors  $\mathbf{x}$  and  $\mathbf{y}$ , and by fixing  $\mathbf{y}$ -variables at some trial values  $\bar{\mathbf{y}}$ , it obtains a *subproblem* which contains only  $\mathbf{x}$ -variables. If the solution to the

subproblem asserts that  $\bar{\mathbf{y}}$  is unacceptable then this solution is used to create a *no-good* constraint which eliminates  $\bar{\mathbf{y}}$  – and possibly other values of  $\mathbf{y}$  – from the search space. Then, this constraint is included in the *master problem* which obtains the next set of values for  $\mathbf{y}$ . This process continues iteratively until the master problem identifies a vector  $\bar{\mathbf{y}}$  which is not rejected by the subproblem. In case the  $\mathbf{x}$ -variables appear in the objective function, the master problem contains a variable  $\eta$  which represents a lower bound on the objective function value of the original problem. The no-good constraint takes the form  $\eta \geq \beta_{\bar{\mathbf{y}}}(\mathbf{y})$ , where  $\beta_{\bar{\mathbf{y}}}(\mathbf{y})$  is a lower bound on the optimal objective function value of the subproblem for any value of  $\mathbf{y}$ . The subscript  $\bar{\mathbf{y}}$  denotes the fixed value of  $\mathbf{y}$  which resulted in this bounding function.

Unlike the classical Benders, in which the subproblems are always continuous and cuts are created by LP or Lagrangian duality, LBBDD is based on the *inference duality* concept and the subproblems may be arbitrary optimization problems. The inference dual is to deduce the tightest bound from the constraints of the problem. However, there is no standard way of obtaining the logic-based Benders cuts and they must be tailored to each problem type. Nevertheless, a valid bounding function  $\beta_{\bar{\mathbf{y}}}(\mathbf{y})$  must satisfy the following two properties:

**Property 4.1.**  $f(\mathbf{x}, \mathbf{y}) \geq \beta_{\bar{\mathbf{y}}}(\mathbf{y})$  for any feasible  $(\mathbf{x}, \mathbf{y})$ , where  $f(\mathbf{x}, \mathbf{y})$  is the objective function of the problem.

**Property 4.2.**  $\beta_{\bar{\mathbf{y}}}(\bar{\mathbf{y}}) = \beta$ , where  $\beta$  is the optimal objective function value of the subproblem obtained by fixing  $\mathbf{y}$  to  $\bar{\mathbf{y}}$ .

The following result is due to [Hooker \(2000\)](#).

**Theorem 4.3.** *If the bounding function  $\beta_{\bar{\mathbf{y}}}(\mathbf{y})$  satisfies Properties 4.1 and 4.2 in each iteration of the Benders algorithm, and the domain of  $\mathbf{y}$  is finite, then the Benders algorithm converges to the optimal value of the problem after finitely many steps.*

In this regard, one of the main contribution of this chapter is to find the exact form of the bounding function for a scheduling problem with a non-regular minsum performance measure – i.e., *Rm-UCDD*.

### 4.3.2 LBB for $Rm$ -UCDD

As already noted couple of times up to this point, we need to develop custom logic-based Benders cuts for our problem. To this end, we start out with two simple, yet powerful observations that (i) once the job-to-machine assignments are fixed, the problem decomposes into single-machine scheduling problems and (ii) even though the resulting problems are still  $\mathcal{NP}$ -hard, they all have a V-shaped schedule without a straddling job (Hall and Posner, 1991).

The second observation suggests us to separate the machine assignment decisions into two parts so that jobs are directly assigned to the early or tardy side of a machine. That way, we obtain two subproblems for each machine. Moreover, the solutions of the resulting subproblems may be calculated in polynomial time because the V-shaped property prescribes that the jobs assigned to the early and tardy sides of each machine are sequenced according to the weighted longest processing time (WLPT) and weighted shortest processing time (WSPT) rules of Smith (1956), respectively. The idea of separating the assignments to the early and tardy sides of the machines is recently used by Plateau and Rios-Solis (2010) for the same problem as ours and by Alvarez-Valdes et al. (2012) for a single-machine UCDD problem while formulating their problems as CQIPs.

Motivated by these observations, we use the same binary variables  $y_{kj}^E$  and  $y_{kj}^T$  introduced at the beginning of this section and define additional auxiliary variables  $\eta_k^E$  and  $\eta_k^T$  which represent the lower bounds on the total costs charged against the jobs performed on the early and tardy sides of machine  $k$ , respectively. Therefore,  $Rm$ -UCDD may be formulated in the LBB framework as follows:

$$(LBF) \quad \text{minimize} \quad \sum_{k=1}^m \eta_k^E + \eta_k^T \quad (4.4)$$

$$\text{subject to} \quad \sum_{k=1}^m y_{kj}^E + y_{kj}^T = 1, \quad j = 1, \dots, n, \quad (4.5)$$

$$\eta_k^E \geq \beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\mathbf{y}_k^E), \quad \bar{\mathbf{y}}_k^E \in \{0, 1\}^n, \quad k = 1, \dots, m, \quad (4.6)$$

$$\eta_k^T \geq \beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T), \quad \bar{\mathbf{y}}_k^T \in \{0, 1\}^n, \quad k = 1, \dots, m, \quad (4.7)$$

$$\eta_k^E \geq 0, \quad \eta_k^T \geq 0 \quad k = 1, \dots, m, \quad (4.8)$$

$$y_{kj}^E \in \{0, 1\}, \quad y_{kj}^T \in \{0, 1\} \quad j = 1, \dots, n, \quad k = 1, \dots, m, \quad (4.9)$$

where  $\beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\mathbf{y}_k^E)$  and  $\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T)$  are the bounding functions on the total weighted earliness and tardiness of the jobs assigned to machine  $k$ , respectively. The subscripts  $\bar{\mathbf{y}}_k^E$  and  $\bar{\mathbf{y}}_k^T$  denote possible fixed values of  $\mathbf{y}_k^E = \{y_{kj}^E \mid j = 1, \dots, n\}$  and  $\mathbf{y}_k^T = \{y_{kj}^T \mid j = 1, \dots, n\}$  which give rise to  $\beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\mathbf{y}_k^E)$  and  $\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T)$ , respectively. Note that in (4.6) and (4.7), a Benders cut for each side of every machine is generated for every possible job to machine assignment – i.e.,  $\bar{\mathbf{y}}_k^E \in \{0, 1\}^n$  and  $\bar{\mathbf{y}}_k^T \in \{0, 1\}^n$ , respectively. Thus, as long as the bounding functions are valid, the sets of constraints (4.6) and (4.7) collectively ensure that the cost of every possible assignment is calculated correctly. The job partitioning constraints (4.5) mandate that each job is assigned to exactly one side of a machine. Thus, **(LBF)** is an exact formulation for *Rm-UCDD* in the LBBDD framework. However, due to the sheer number of constraints present in the model, solving the monolithic formulation of **(LBF)** is not a viable option. To be specific, there are  $2^{n+1}m + n$  constraints in the formulation and even for a modest size instance with 75 jobs and 5 machines, the number of constraints is  $3.8 \times 10^{23}$  – which is in the order of the number of stars in the observable universe. Therefore, a delayed constraint generation scheme is proposed to solve **(LBF)** and the pseudo-code of the LBBDD algorithm is stated in Algorithm 6 at the end of this section.

We next turn our attention to the identification of the bounding functions. First, we focus on the bounding function of the tardy side and develop a strengthened logic-based Benders cut. Then, the bounding function of the early side follows by a similar argument and it is presented at the end of this section. Let  $\bar{\mathbf{y}}^T$  be a fixed job to tardy side assignment and  $\bar{\mathbf{y}}_k^T$  represent its  $k$ th column – i.e., only those related to machine  $k$  – and  $J(\bar{\mathbf{y}}_k^T) = \{j \mid \bar{y}_{kj}^T = 1\}$  be the set of jobs assigned to this side of machine  $k$ . Then, the minimum total weighted tardiness on machine  $k$  is equal to

$$z_k^T(\bar{\mathbf{y}}_k^T) = \sum_{j \in J(\bar{\mathbf{y}}_k^T)} \pi_j T_j(\bar{\mathbf{y}}_k^T), \quad \text{where} \quad T_j(\bar{\mathbf{y}}_k^T) = p_{jk} + \sum_{i \in J(\bar{\mathbf{y}}_k^T), i <_k^T j} p_{ik}. \quad (4.10)$$

The most obvious bounding function would be of the form:

$$\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T) = z_k^T(\bar{\mathbf{y}}_k^T) (1 - I_k^T), \quad \text{where} \quad I_k^T = |J(\bar{\mathbf{y}}_k^T)| - \sum_{j \in J(\bar{\mathbf{y}}_k^T)} y_{kj}^T. \quad (4.11)$$

This bounding function trivially satisfies Properties 4.1 and 4.2; however, it is unnecessarily weak due to the fact that it eliminates only the solution that give rise to the bound and any change to the assignment of the jobs in  $J(\bar{\mathbf{y}}_k^T)$  renders the bound redundant. A prevalent method used in the literature to strengthen the bounding functions is to find a smaller set of jobs that results in the same solution to the scheduling subproblem. In such studies (Ciré et al., 2015, Coban and Hooker, 2013, Hooker, 2004, 2005a,b, 2006, 2007b, Hooker and Ottosson, 2003), either the subproblem is only a feasibility problem and the authors can determine a subset of jobs that still leads to the infeasibility, or they make use of the fact that there are non-zero release dates in the problem and some of the jobs can be removed without decreasing the objective function value. Unfortunately, this is not possible in our case because the subproblem is an optimization problem which is always feasible and all jobs are ready at time zero; and thus, removing any job would decrease the objective function value of the subproblem. Nevertheless, the bounding functions developed in the remainder of this section are very strong and the results of the computational study, presented in the next section, attest to the quality of the proposed logic-based Benders cuts created with these bounding functions.

### 4.3.3 Strengthened Bounding Functions

A stronger bounding function  $\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T)$  can be obtained by taking into account the actual contributions made to the objective function value by the individual jobs. This yields the bounding function

$$\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T) = z_k^T(\bar{\mathbf{y}}_k^T) - \sum_{j \in J(\bar{\mathbf{y}}_k^T)} \left( \pi_j T_j(\bar{\mathbf{y}}_k^T) + p_{jk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), j <_k^T i} \pi_i \right) (1 - y_{kj}^T), \quad (4.12)$$

which is based on the fact that if job  $j \in J(\bar{\mathbf{y}}_k^T)$  were to be removed from machine  $k$  then the total tardiness of the jobs that are scheduled after job  $j$  would decrease by at most  $p_{jk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), j <_k^T i} \pi_i$  units. Thus, this function also satisfies both properties as well.

Another venue of improvement arises by studying the potential contribution of the jobs that are currently not assigned to this side of machine  $k$ . That is, if job



$l \notin J(\bar{\mathbf{y}}_k^T)$  were to be assigned to the tardy side of machine  $k$ , then this job would incur a cost of  $\pi_l(p_{lk} + \sum_{i \in J(\bar{\mathbf{y}}_k^T), i <_k^T l} p_{ik})$  and the total tardiness of the jobs that need to be scheduled after job  $l$  would increase by  $p_{lk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), l <_k^T i} \pi_i$  units. It is a simple matter to show that as long as the set of jobs  $J(\bar{\mathbf{y}}_k^T)$  stays on the tardy side of machine  $k$ , these cost calculations are valid. This observation yields the following bounding function

$$\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T) = z_k^T(\bar{\mathbf{y}}_k^T) (1 + I_k^T) + \sum_{l \notin J(\bar{\mathbf{y}}_k^T)} \left( \pi_l T_l(\bar{\mathbf{y}}_k^T) + p_{lk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), l <_k^T i} \pi_i \right) (y_{kl}^T - I_k^T), \quad (4.13)$$

where  $T_l(\bar{\mathbf{y}}_k^T)$  is given in (4.10). Unfortunately, this bounding function has the same drawback as that of (4.11) – i.e., it is unnecessarily weak and setting even one  $y_{kj}^T, j \in J(\bar{\mathbf{y}}_k^T)$  to zero renders the bound redundant.

Note that we cannot strengthen (4.12) by using the same reasoning we have used to obtain (4.13) from (4.11). First, the resulting bounding function would still suffer from the same phenomenon that changing the assignment of a single job in  $J(\bar{\mathbf{y}}_k^T)$  renders the additional term useless. Secondly and more importantly, the additional term would weaken the bounding function when there is more than one assignment change in the set  $J(\bar{\mathbf{y}}_k^T)$ . Nevertheless, it is still possible to incorporate the tardiness costs of the jobs that are not in the set  $J(\bar{\mathbf{y}}_k^T)$ . The reasoning is similar to that which led us to (4.12). That is, we need to consider the contribution of each additional job individually; however, this is not very straightforward due to the cost difference resulting from the interaction between the jobs that are added to and removed from this side of the machine. By analyzing the structure of the subproblem solutions, we determine the terms that need to be subtracted from the coefficients of  $(1 - y_{kj}^T)$  and  $y_{kl}^T$ , for  $j \in J(\bar{\mathbf{y}}_k^T)$  and  $l \notin J(\bar{\mathbf{y}}_k^T)$  in order to obtain a valid bounding function of the following form:

$$\begin{aligned} \beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T) = & z_k^T(\bar{\mathbf{y}}_k^T) - \sum_{j \in J(\bar{\mathbf{y}}_k^T)} \left( \pi_j \left( T_j(\bar{\mathbf{y}}_k^T) + \frac{p_{jk}}{2} \right) + p_{jk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), j <_k^T i} \pi_i \right) (1 - y_{kj}^T) \\ & + \sum_{l \notin J(\bar{\mathbf{y}}_k^T)} \left( \pi_l \left( T_l(\bar{\mathbf{y}}_k^T) - \frac{p_{lk}}{2} \right) + p_{lk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), l <_k^T i} \pi_i \right) y_{kl}^T, \end{aligned} \quad (4.14)$$

where  $z_k^T(\bar{\mathbf{y}}_k^T)$ ,  $T_j(\bar{\mathbf{y}}_k^T)$  are defined in (4.10). Note that Property 4.2 holds trivially for (4.14); however, we need to show that Property 4.1 is satisfied as well. To this

end, we first need the following somewhat more general result.

**Lemma 4.4.** *Let  $a_j$  and  $b_j$ ,  $j \in D$  be nonnegative real numbers where  $D$  is an arbitrary index set. Let  $A$  and  $R$  be two disjoint subsets of  $D$  such that  $A \cup R = D$  and let  $S_j$  and  $S_j^c$  denote the sets which includes index  $j \in D$  and its complement, respectively. That is,  $S_j = \{S \in \{A, R\} \mid j \in S\}$  and  $S_j^c = \{S \in \{A, R\} \mid j \notin S\}$ . Finally, the notation  $i < j$  indicates that either  $\frac{a_i}{b_i} > \frac{a_j}{b_j}$  or  $\frac{a_i}{b_i} = \frac{a_j}{b_j}$  and  $i < j$  where  $i, j \in D$ , and the notation  $[l] \in D$ ,  $l = 1, \dots, |D|$ , denotes the  $l$ th index based on the  $<$  ordering of the indices in  $D$ . Then,*

$$\sum_{j \in D} a_j \left( b_j + 2 \sum_{i \in S_j, i < j} b_i - 2 \sum_{i \in S_j^c, i < j} b_i \right) \geq \frac{a_{[l]}}{b_{[l]}} \left( b_{[l]} + \sum_{i \in S_{[l]}, i < [l]} b_i - \sum_{i \in S_{[l]}^c, i < [l]} b_i \right)^2 + \sum_{j \in D, [l] < j} a_j \left( b_j + 2 \sum_{i \in S_j, i < j} b_i - 2 \sum_{i \in S_j^c, i < j} b_i \right) \quad (4.15)$$

holds for  $l = 1, \dots, |D|$ .

*Proof.* We first rearrange the terms of (4.15) and obtain the following:

$$\sum_{j \in D, j < [l]} a_j \left( b_j + 2 \sum_{i \in S_j, i < j} b_i - 2 \sum_{i \in S_j^c, i < j} b_i \right) + a_{[l]} \left( b_{[l]} + 2 \sum_{i \in S_{[l]}, i < [l]} b_i - 2 \sum_{i \in S_{[l]}^c, i < [l]} b_i \right) \geq \frac{a_{[l]}}{b_{[l]}} \left( b_{[l]} + \sum_{i \in S_{[l]}, i < [l]} b_i - \sum_{i \in S_{[l]}^c, i < [l]} b_i \right)^2. \quad (4.16)$$

The rest of the proof is then by induction and proceeds as follows.

**Base case:** When  $l = 1$ , both sides of (4.16) equal to  $a_{[1]}b_{[1]}$  since  $\{j \in D \mid j < [1]\} = \emptyset$ .

Thus, (4.15) is true for  $l = 1$ .

**Induction step:** Let  $u \in \{1, \dots, |D| - 1\}$  be given and suppose (4.15) – thus, (4.16)

– holds for  $l = u$ . Then, for  $l = u + 1$

$$\sum_{\substack{j \in D \\ j < [u+1]}} a_j \left( b_j + 2 \sum_{\substack{i \in S_j \\ i < j}} b_i - 2 \sum_{\substack{i \in S_j^c \\ i < j}} b_i \right) + a_{[u+1]} \left( b_{[u+1]} + 2 \sum_{\substack{i \in S_{[u+1]} \\ i < [u+1]}} b_i - 2 \sum_{\substack{i \in S_{[u+1]}^c \\ i < [u+1]}} b_i \right) \quad (4.17a)$$

$$= \sum_{j \in D, j < [u]} a_j \left( b_j + 2 \sum_{i \in S_j, i < j} b_i - 2 \sum_{i \in S_j^c, i < j} b_i \right) + a_{[u]} \left( b_{[u]} + 2 \sum_{i \in S_{[u]}, i < [u]} b_i - 2 \sum_{i \in S_{[u]}^c, i < [u]} b_i \right) + \frac{a_{[u+1]}}{b_{[u+1]}} \left( (b_{[u+1]})^2 + 2b_{[u+1]} \left( \sum_{\substack{i \in S_{[u+1]} \\ i < [u+1]}} b_i - \sum_{\substack{i \in S_{[u+1]}^c \\ i < [u+1]}} b_i \right) \right) \quad (4.17b)$$

$$\geq \frac{a_{[u]}}{b_{[u]}} \left( b_{[u]} + \sum_{\substack{i \in S_{[u]} \\ i < [u]}} b_i - \sum_{\substack{i \in S_{[u]}^c \\ i < [u]}} b_i \right)^2 + \frac{a_{[u+1]}}{b_{[u+1]}} \left( (b_{[u+1]})^2 + 2b_{[u+1]} \left( \sum_{\substack{i \in S_{[u+1]} \\ i < [u+1]}} b_i - \sum_{\substack{i \in S_{[u+1]}^c \\ i < [u+1]}} b_i \right) \right) \quad (4.17c)$$

$$\geq \frac{a_{[u+1]}}{b_{[u+1]}} \left( \sum_{\substack{i \in S_{[u+1]} \\ i < [u+1]}} b_i - \sum_{\substack{i \in S_{[u+1]}^c \\ i < [u+1]}} b_i \right)^2 + \frac{a_{[u+1]}}{b_{[u+1]}} \left( (b_{[u+1]})^2 + 2b_{[u+1]} \left( \sum_{\substack{i \in S_{[u+1]} \\ i < [u+1]}} b_i - \sum_{\substack{i \in S_{[u+1]}^c \\ i < [u+1]}} b_i \right) \right) \quad (4.17d)$$

$$= \frac{a_{[u+1]}}{b_{[u+1]}} \left( b_{[u+1]} + \sum_{\substack{i \in S_{[u+1]} \\ i < [u+1]}} b_i - \sum_{\substack{i \in S_{[u+1]}^c \\ i < [u+1]}} b_i \right)^2. \quad (4.17e)$$

Transition from (4.17b) to (4.17c) is due to the induction hypothesis, and that from (4.17c) to (4.17d) is due to the fact that  $\frac{a_{[u]}}{b_{[u]}} \geq \frac{a_{[u+1]}}{b_{[u+1]}}$ . The remaining steps are due to rearrangements and basic transformations. Therefore, (4.17) establishes the correctness of (4.16) for  $l = u + 1$ , and by the principle of induction, (4.16) – thus, (4.15) – holds for all  $l \in \{1, \dots, |D|\}$ .  $\square$

Before proving the validity of the proposed bounding function we need one more result. To this end, let  $\bar{\mathbf{y}}_k^T$  and  $\bar{\mathbf{y}}_k^{T'}$  be two different job partitions on the tardy side of machine  $k$  with total costs  $z_k^T(\bar{\mathbf{y}}_k^T)$  and  $z_k^T(\bar{\mathbf{y}}_k^{T'})$ , respectively. Using the same notation as in Lemma 4.4, let  $A$  and  $R$  denote the sets of jobs that need to be added to and removed from  $J(\bar{\mathbf{y}}_k^T)$  to obtain  $J(\bar{\mathbf{y}}_k^{T'})$ , respectively. That is,  $A = J(\bar{\mathbf{y}}_k^{T'}) \setminus J(\bar{\mathbf{y}}_k^T)$  and  $R = J(\bar{\mathbf{y}}_k^T) \setminus J(\bar{\mathbf{y}}_k^{T'})$ . Furthermore, let  $D$  denote the symmetric difference between sets  $J(\bar{\mathbf{y}}_k^T)$  and  $J(\bar{\mathbf{y}}_k^{T'})$  – i.e.,  $D = J(\bar{\mathbf{y}}_k^T) \Delta J(\bar{\mathbf{y}}_k^{T'})$ . Note that the index sets  $A$  and  $R$  are disjoint by definition and  $D = A \cup R$ . Finally, let  $a_j = \pi_j$  and  $b_j = p_{jk}$  for  $j \in D$ , and let  $<_k^T$  be the associated precedence relation and  $[l] \in D, l = 1, \dots, |D|$ , denote the  $l$ th job based on the  $<_k^T$  ordering of the jobs in  $D$ . Then, a direct corollary of Lemma 4.4 is as follows:

**Corollary 4.5.** *The following inequality holds for  $l = 1, \dots, |D|$ :*

$$\begin{aligned} \sum_{j \in D} \pi_j \left( p_{jk} + 2 \sum_{i \in S_j, i <_k^T j} p_{ik} - 2 \sum_{i \in S_j^c, i <_k^T j} p_{ik} \right) &\geq \frac{\pi_{[l]}}{p_{[l]k}} \left( p_{[l]k} + \sum_{i \in S_{[l]}, i <_k^T [l]} p_{ik} - \sum_{i \in S_{[l]}^c, i <_k^T [l]} p_{ik} \right)^2 \\ &+ \sum_{j \in D, [l] <_k^T j} \pi_j \left( p_{jk} + 2 \sum_{i \in S_j, i <_k^T j} p_{ik} - 2 \sum_{i \in S_j^c, i <_k^T j} p_{ik} \right). \end{aligned} \quad (4.18)$$

We now prove the validity of the proposed bounding function.

**Proposition 4.6.** *The bounding function  $\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T)$  given in (4.14) satisfies Property 4.1.*

*Proof.* As alluded to earlier, we need to show that the bounding function provides a lower bound for the actual total tardiness on machine  $k$ . More specifically, we need to show that the following inequality holds for all combinations of fixed job assignments on the tardy side of machine  $k$  – i.e.,  $\bar{\mathbf{y}}_k^T$  and  $\bar{\mathbf{y}}_k^{T'}$ :

$$0 \leq z_k^T(\bar{\mathbf{y}}_k^{T'}) - \beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\bar{\mathbf{y}}_k^{T'}). \quad (4.19)$$

Note that  $\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\bar{\mathbf{y}}_k^T)$  is calculated with the bounding function obtained for  $\bar{\mathbf{y}}_k^T$ . Whereas,  $z_k^T(\bar{\mathbf{y}}_k^T)$  denotes the actual objective function value for  $\bar{\mathbf{y}}_k^T$ . For ease of perusal, each term is expanded individually and merged back together.

Using the same notation as in Corollary 4.5,  $\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\bar{\mathbf{y}}_k^T)$  expands as follows:

$$\begin{aligned}
\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\bar{\mathbf{y}}_k^T) &= z_k^T(\bar{\mathbf{y}}_k^T) - \sum_{j \in R} \left( \pi_j \left( T_j(\bar{\mathbf{y}}_k^T) + \frac{p_{jk}}{2} \right) + p_{jk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), j <_k^T i} \pi_i \right) \\
&\quad + \sum_{l \in A} \left( \pi_l \left( T_l(\bar{\mathbf{y}}_k^T) - \frac{p_{lk}}{2} \right) + p_{lk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), l <_k^T i} \pi_i \right) \\
&= \sum_{j \in J(\bar{\mathbf{y}}_k^T)} \pi_j T_j(\bar{\mathbf{y}}_k^T) - \sum_{j \in R} p_{jk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), j <_k^T i} \pi_i + \sum_{l \in A} p_{lk} \sum_{i \in J(\bar{\mathbf{y}}_k^T), l <_k^T i} \pi_i - \sum_{j \in R} \pi_j \left( T_j(\bar{\mathbf{y}}_k^T) + \frac{p_{jk}}{2} \right) \\
&\quad + \sum_{l \in A} \pi_l \left( T_l(\bar{\mathbf{y}}_k^T) - \frac{p_{lk}}{2} \right) \\
&= \sum_{j \in J(\bar{\mathbf{y}}_k^T)} \pi_j \left( T_j(\bar{\mathbf{y}}_k^T) - \sum_{i \in R, i <_k^T j} p_{ik} + \sum_{i \in A, i <_k^T j} p_{ik} \right) - \sum_{j \in R} \pi_j \left( T_j(\bar{\mathbf{y}}_k^T) + \frac{p_{jk}}{2} \right) \\
&\quad + \sum_{l \in A} \pi_l \left( T_l(\bar{\mathbf{y}}_k^T) - \frac{p_{lk}}{2} \right). \tag{4.20}
\end{aligned}$$

Similarly, the objective function value associated with the tardy side of machine  $k$  for the fixed job to machine assignment  $\bar{\mathbf{y}}_k^T$  is

$$\begin{aligned}
z_k^T(\bar{\mathbf{y}}_k^T) &= \sum_{j \in J(\bar{\mathbf{y}}_k^T)} \pi_j T_j(\bar{\mathbf{y}}_k^T) = \sum_{j \in J(\bar{\mathbf{y}}_k^T)} \pi_j T_j(\bar{\mathbf{y}}_k^T) - \sum_{j \in R} \pi_j T_j(\bar{\mathbf{y}}_k^T) + \sum_{l \in A} \pi_l T_l(\bar{\mathbf{y}}_k^T) \\
&= \sum_{j \in J(\bar{\mathbf{y}}_k^T)} \pi_j \left( T_j(\bar{\mathbf{y}}_k^T) - \sum_{i \in R, i <_k^T j} p_{ik} + \sum_{i \in A, i <_k^T j} p_{ik} \right) - \sum_{j \in R} \pi_j \left( T_j(\bar{\mathbf{y}}_k^T) - \sum_{i \in R, i <_k^T j} p_{ik} + \sum_{i \in A, i <_k^T j} p_{ik} \right) \\
&\quad + \sum_{l \in A} \pi_l \left( T_l(\bar{\mathbf{y}}_k^T) - \sum_{i \in R, i <_k^T l} p_{ik} + \sum_{i \in A, i <_k^T l} p_{ik} \right). \tag{4.21}
\end{aligned}$$

Substituting  $\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\bar{\mathbf{y}}_k^T)$  with (4.20) and  $z_k^T(\bar{\mathbf{y}}_k^T)$  with (4.21) in (4.19) and rearranging the terms yields

$$\begin{aligned}
z_k^T(\bar{\mathbf{y}}_k^T) - \beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\bar{\mathbf{y}}_k^T) &= \sum_{l \in A} \pi_l \left( \frac{p_{lk}}{2} + \sum_{i \in A, i <_k^T l} p_{ik} - \sum_{i \in R, i <_k^T l} p_{ik} \right) + \sum_{j \in R} \pi_j \left( \frac{p_{jk}}{2} + \sum_{i \in R, i <_k^T j} p_{ik} - \sum_{i \in A, i <_k^T j} p_{ik} \right) \\
&= \sum_{j \in D} \pi_j \left( \frac{p_{jk}}{2} + \sum_{i \in S_j, i <_k^T j} p_{ik} - \sum_{i \in S_j^c, i <_k^T j} p_{ik} \right) = \frac{1}{2} \sum_{j \in D} \pi_j \left( p_{jk} + 2 \sum_{i \in S_j, i <_k^T j} p_{ik} - 2 \sum_{i \in S_j^c, i <_k^T j} p_{ik} \right) \tag{4.22}
\end{aligned}$$

$$\geq \frac{1}{2} \frac{\pi_{[|D|]}}{p_{[|D|]k}} \left( p_{[|D|]k} + \sum_{\substack{i \in S_{[|D|]} \\ i <_k^T [D]}} p_{ik} - \sum_{\substack{i \in S_{[|D|]}^c \\ i <_k^T [D]}} p_{ik} \right)^2 \geq 0. \quad (4.23)$$

Transition from (4.22) to (4.23) is due to Corollary 4.5, and the non-negativity of (4.23) follows from the fact that  $x^2 \geq 0$  for all  $x \in \mathbb{R}$  and that  $\pi_j, p_{jk} \in \mathbb{R}_{>0}$  for all  $j = 1, \dots, n, k = 1, \dots, m$ . This completes the proof since (4.19) is valid for all combinations of fixed  $\bar{\mathbf{y}}_k^T$  and  $\bar{\mathbf{y}}_k^{T'}$ .  $\square$

Therefore, we can produce a valid logic-based Benders cut of the form

$$\eta_k^T \geq \beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T) \quad (4.24)$$

for the tardy side of machine  $k$  from a fixed job to machine assignment  $\bar{\mathbf{y}}_k^T$ , where the bounding function  $\beta_{\bar{\mathbf{y}}_k^T}^{Tk}(\mathbf{y}_k^T)$  is given in (4.14).

A similar reasoning yields the following bounding function  $\beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\mathbf{y}_k^E)$  for the early side of machine  $k$  from a fixed job to machine assignment  $\bar{\mathbf{y}}_k^E$ :

$$\begin{aligned} \beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\mathbf{y}_k^E) &= z_k^E(\bar{\mathbf{y}}_k^E) - \sum_{j \in J(\bar{\mathbf{y}}_k^E)} \left( \epsilon_j \left( E_j(\bar{\mathbf{y}}_k^E) + \frac{p_{jk}}{2} \right) + p_{jk} \sum_{i \in J(\bar{\mathbf{y}}_k^E), j <_k^E i} \epsilon_i \right) (1 - y_{kj}^E) \\ &\quad + \sum_{l \notin J(\bar{\mathbf{y}}_k^E)} \left( \epsilon_l \left( E_l(\bar{\mathbf{y}}_k^E) - \frac{p_{lk}}{2} \right) + p_{lk} \sum_{i \in J(\bar{\mathbf{y}}_k^E), l <_k^E i} \epsilon_i \right) y_{kl}^E \end{aligned} \quad (4.25)$$

with

$$z_k^E(\bar{\mathbf{y}}_k^E) = \sum_{j \in J(\bar{\mathbf{y}}_k^E)} \epsilon_j E_j(\bar{\mathbf{y}}_k^E), \quad \text{where } E_j(\bar{\mathbf{y}}_k^E) = \sum_{i \in J(\bar{\mathbf{y}}_k^E), i <_k^E j} p_{ik}, \quad (4.26)$$

and  $J(\bar{\mathbf{y}}_k^E) = \{j \mid \bar{y}_{kj}^E = 1\}$ . Property 4.2 is trivially satisfied since for  $\mathbf{y}_k^E = \bar{\mathbf{y}}_k^E$ ,  $\beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\mathbf{y}_k^E) = z_k^E(\bar{\mathbf{y}}_k^E) = z_k^E(\mathbf{y}_k^E)$ . However, to prove the validity of the proposed bounding function we again need to derive a result similar to Corollary 4.7.

To this end, let  $\bar{\mathbf{y}}_k^E$  and  $\bar{\mathbf{y}}_k^{E'}$  be two different job partitions on the early side of machine  $k$  with total costs  $z_k^E(\bar{\mathbf{y}}_k^E)$  and  $z_k^E(\bar{\mathbf{y}}_k^{E'})$ , respectively. Using the notation of Lemma 4.4, let  $A$  and  $R$  denote the sets of jobs that need to be added to and removed from  $J(\bar{\mathbf{y}}_k^E)$  to obtain  $J(\bar{\mathbf{y}}_k^{E'})$ , respectively. That is,  $A = J(\bar{\mathbf{y}}_k^{E'}) \setminus J(\bar{\mathbf{y}}_k^E)$  and  $R = J(\bar{\mathbf{y}}_k^E) \setminus J(\bar{\mathbf{y}}_k^{E'})$ . Furthermore, let  $D$  denote the symmetric difference between sets  $J(\bar{\mathbf{y}}_k^E)$  and  $J(\bar{\mathbf{y}}_k^{E'})$ —i.e.,  $D = J(\bar{\mathbf{y}}_k^E) \Delta J(\bar{\mathbf{y}}_k^{E'})$ . Note that the index sets  $A$  and  $R$  are

disjoint by definition and  $D = A \cup R$ . Finally, let  $a_j = \epsilon_j$  and  $b_j = p_{jk}$  for  $j \in D$ , and let  $\prec_k^E$  be the associated precedence notation and  $[l] \in D, l = 1, \dots, |D|$ , denote the  $l$ th job based on  $\prec_k^E$  ordering of the jobs in  $D$ . Then, a direct corollary of Lemma 4.4 is as follows:

**Corollary 4.7.** *The following inequality holds for  $l = 1, \dots, |D|$ :*

$$\begin{aligned} \sum_{j \in D} \epsilon_j \left( p_{jk} + 2 \sum_{i \in S_j, i \prec_k^E j} p_{ik} - 2 \sum_{i \in S_j^c, i \prec_k^E j} p_{ik} \right) &\geq \frac{\epsilon_{[l]}}{p_{[l]k}} \left( p_{[l]k} + \sum_{i \in S_{[l]}, i \prec_k^E [l]} p_{ik} - \sum_{i \in S_{[l]}^c, i \prec_k^E [l]} p_{ik} \right)^2 \\ &+ \sum_{j \in D, [l] \prec_k^E j} \epsilon_j \left( p_{jk} + 2 \sum_{i \in S_j, i \prec_k^E j} p_{ik} - 2 \sum_{i \in S_j^c, i \prec_k^E j} p_{ik} \right). \end{aligned} \quad (4.27)$$

We now prove the validity of the proposed bounding function.

**Proposition 4.8.** *The bounding function  $\beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\mathbf{y}_k^E)$  given in (4.25) satisfies Property 4.1.*

*Proof.* The proof follows very closely that of Proposition 4.6 and we show that the bounding function provides a lower bound for the actual total earliness on machine  $k$ . More specifically, we need to show that the following inequality holds for all combinations of fixed job assignments on the early side of machine  $k$  – i.e.,  $\bar{\mathbf{y}}_k^E$  and  $\bar{\mathbf{y}}_k^{E'}$ :

$$0 \leq z_k^E(\bar{\mathbf{y}}_k^{E'}) - \beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\bar{\mathbf{y}}_k^{E'}). \quad (4.28)$$

Note that  $\beta_{\bar{\mathbf{y}}_k^E}^{Tk}(\bar{\mathbf{y}}_k^{E'})$  is calculated with the bounding function obtained for  $\bar{\mathbf{y}}_k^E$ . Whereas,  $z_k^E(\bar{\mathbf{y}}_k^{E'})$  denotes the actual objective function value for  $\bar{\mathbf{y}}_k^{E'}$ . For ease of perusal, each term is expanded individually and merged back together.

Using the same notation as in Corollary 4.5,  $\beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\bar{\mathbf{y}}_k^{E'})$  expands as follows:

$$\begin{aligned} \beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\bar{\mathbf{y}}_k^{E'}) &= z_k^E(\bar{\mathbf{y}}_k^E) - \sum_{j \in R} \left( \epsilon_j \left( T_j(\bar{\mathbf{y}}_k^T) + \frac{p_{jk}}{2} \right) + p_{jk} \sum_{i \in J(\bar{\mathbf{y}}_k^E), j \prec_k^E i} \epsilon_i \right) + \sum_{l \in A} \left( \epsilon_l \left( T_l(\bar{\mathbf{y}}_k^T) - \frac{p_{lk}}{2} \right) + p_{lk} \sum_{i \in J(\bar{\mathbf{y}}_k^E), l \prec_k^E i} \epsilon_i \right) \\ &= \sum_{j \in J(\bar{\mathbf{y}}_k^E)} \epsilon_j T_j(\bar{\mathbf{y}}_k^T) - \sum_{j \in R} p_{jk} \sum_{i \in J(\bar{\mathbf{y}}_k^E), j \prec_k^E i} \epsilon_i + \sum_{l \in A} p_{lk} \sum_{i \in J(\bar{\mathbf{y}}_k^E), l \prec_k^E i} \epsilon_i - \sum_{j \in R} \epsilon_j \left( T_j(\bar{\mathbf{y}}_k^T) + \frac{p_{jk}}{2} \right) \\ &\quad + \sum_{l \in A} \epsilon_l \left( T_l(\bar{\mathbf{y}}_k^T) - \frac{p_{lk}}{2} \right) \\ &= \sum_{j \in J(\bar{\mathbf{y}}_k^E)} \epsilon_j \left( T_j(\bar{\mathbf{y}}_k^T) - \sum_{i \in R, i \prec_k^E j} p_{ik} + \sum_{i \in A, i \prec_k^E j} p_{ik} \right) - \sum_{j \in R} \epsilon_j \left( T_j(\bar{\mathbf{y}}_k^T) + \frac{p_{jk}}{2} \right) \\ &\quad + \sum_{l \in A} \epsilon_l \left( T_l(\bar{\mathbf{y}}_k^T) - \frac{p_{lk}}{2} \right). \end{aligned} \quad (4.29)$$

Similarly, the objective function value associated with the early side of machine  $k$  for the fixed job to machine assignment  $\bar{\mathbf{y}}_k^{E'}$  is

$$\begin{aligned}
z_k^E(\bar{\mathbf{y}}_k^{E'}) &= \sum_{j \in J(\bar{\mathbf{y}}_k^{E'})} \epsilon_j T_j(\bar{\mathbf{y}}_k^{E'}) = \sum_{j \in J(\bar{\mathbf{y}}_k^E)} \epsilon_j T_j(\bar{\mathbf{y}}_k^{E'}) - \sum_{j \in R} \epsilon_j T_j(\bar{\mathbf{y}}_k^{E'}) + \sum_{l \in A} \epsilon_l T_l(\bar{\mathbf{y}}_k^{E'}) \\
&= \sum_{j \in J(\bar{\mathbf{y}}_k^E)} \epsilon_j \left( T_j(\bar{\mathbf{y}}_k^T) - \sum_{i \in R, i <_k^E j} p_{ik} + \sum_{i \in A, i <_k^E j} p_{ik} \right) - \sum_{j \in R} \epsilon_j \left( T_j(\bar{\mathbf{y}}_k^T) - \sum_{i \in R, i <_k^E j} p_{ik} + \sum_{i \in A, i <_k^E j} p_{ik} \right) \\
&\quad + \sum_{l \in A} \epsilon_l \left( T_l(\bar{\mathbf{y}}_k^T) - \sum_{i \in R, i <_k^E l} p_{ik} + \sum_{i \in A, i <_k^E l} p_{ik} \right). \quad (4.30)
\end{aligned}$$

Substituting  $\beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\bar{\mathbf{y}}_k^{E'})$  with (4.29) and  $z_k^E(\bar{\mathbf{y}}_k^{E'})$  with (4.30) in (4.28) and rearranging the terms yields

$$\begin{aligned}
z_k^E(\bar{\mathbf{y}}_k^{E'}) - \beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\bar{\mathbf{y}}_k^{E'}) &= \sum_{l \in A} \epsilon_l \left( \frac{p_{lk}}{2} + \sum_{i \in A, i <_k^E l} p_{ik} - \sum_{i \in R, i <_k^E l} p_{ik} \right) + \sum_{j \in R} \epsilon_j \left( \frac{p_{jk}}{2} + \sum_{i \in R, i <_k^E j} p_{ik} - \sum_{i \in A, i <_k^E j} p_{ik} \right) \\
&= \sum_{j \in D} \epsilon_j \left( \frac{p_{jk}}{2} + \sum_{i \in S_j, i <_k^E j} p_{ik} - \sum_{i \in S_j^c, i <_k^E j} p_{ik} \right) = \frac{1}{2} \sum_{j \in D} \epsilon_j \left( p_{jk} + 2 \sum_{i \in S_j, i <_k^E j} p_{ik} - 2 \sum_{i \in S_j^c, i <_k^E j} p_{ik} \right) \quad (4.31) \\
&\geq \frac{1}{2} \frac{\epsilon_{[D]}}{p_{[D]k}} \left( p_{[D]k} + \sum_{\substack{i \in S_{[D]}, \\ i <_k^E [D]}} p_{ik} - \sum_{\substack{i \in S_{[D]}^c, \\ i <_k^E [D]}} p_{ik} \right)^2 \geq 0. \quad (4.32)
\end{aligned}$$

Transition from (4.31) to (4.32) is due to Corollary 4.5, and the non-negativity of (4.32) follows from the fact that  $x^2 \geq 0$  for all  $x \in \mathbb{R}$  and that  $\epsilon_j, p_{jk} \in \mathbb{R}_{>0}$  for all  $j = 1, \dots, n, k = 1, \dots, m$ . This completes the proof since (4.28) is valid for all combinations of fixed  $\bar{\mathbf{y}}_k^E$  and  $\bar{\mathbf{y}}_k^{E'}$ .  $\square$

Consequently, we may generate the following cut from a fixed job assignment on the early side of machine  $k$  – i.e.,  $\bar{\mathbf{y}}_k^E$  – using the bounding function  $\beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\mathbf{y}_k^E)$  given in (4.25):

$$\eta_k^E \geq \beta_{\bar{\mathbf{y}}_k^E}^{Ek}(\mathbf{y}_k^E). \quad (4.33)$$

The pseudo-code of our LBD algorithm for solving (LBF) is stated in Algorithm 6. The correctness of the algorithm is argued through Theorem 4.3, Propositions 4.6 and 4.8, and the finiteness of the feasible assignments.

In classical Benders decomposition applications, the master problem is solved to (near-)optimality, then cuts are generated based on the master problem solution, and then the master problem re-optimized with the additional cuts. This procedure is repeated in a loop until some stopping conditions are satisfied. However, note that this may lead to an unsatisfactory computational performance because a new search tree is constructed in each iteration and the same nodes are explored from scratch again and again. For further information on this matter, the reader is referred to the discussions offered in (Rubin, 2011) and Section 2.4.1 of Chapter 2. Instead, we refrain from the classical textbook application approach and execute our decomposition algorithm on a single search tree using the *lazy constraint callback* feature of IBM ILOG CPLEX (2013). The pseudo-code of our algorithm reflects the use of the lazy constraint technology.

---

**Algorithm 6:** Solving (LBF) by logic-based Benders decomposition and lazy constraint generation.

---

```

1 Create the relaxed master problem (RMP) with (4.4), (4.5), (4.8), and (4.9) ;
  // Initialization.
2 Invoke CPLEX on (RMP); // Main loop.
3 repeat
4   Identify a new candidate incumbent solution  $(\bar{\mathbf{y}}^E, \bar{\mathbf{y}}^T)$  with an objective
   function value of  $\sum_{k=1}^m (\bar{\eta}_k^E + \bar{\eta}_k^T)$ ;
5   for  $k = 1$  to  $m$  do
6     Compute the optimal objective values  $z_k^E(\bar{\mathbf{y}}_k^E)$  and  $z_k^T(\bar{\mathbf{y}}_k^T)$  of the jobs
     assigned to the early and tardy sides of machine  $k$ , respectively;
7     if  $\bar{\eta}_k^E < z_k^E(\bar{\mathbf{y}}_k^E)$  then //  $\bar{\mathbf{y}}_k^E$  violates a missing Benders cut.
8       Generate a logic-based Benders cut of the form (4.33) from  $\bar{\mathbf{y}}_k^E$ ;
9       Add the cut to (RMP) as a lazy constraint;
10    if  $\bar{\eta}_k^T < z_k^T(\bar{\mathbf{y}}_k^T)$  then //  $\bar{\mathbf{y}}_k^T$  violates a missing Benders cut.
11      Generate a logic-based Benders cut of the form (4.24) from  $\bar{\mathbf{y}}_k^T$ ;
12      Add the cut to (RMP) as a lazy constraint;
13 until CPLEX determines that the relative optimality gap of the current incumbent is less
    than some threshold;
14 The best available job partition  $(\bar{\mathbf{y}}^E, \bar{\mathbf{y}}^T)^*$  for (LBF) is retrieved from CPLEX. The
    optimal solution for Rm-UCDD is obtained by applying the WLPT and WSPT rules
    independently to the sets of jobs on the early and tardy sides of each machine,
    respectively;
```

---



## 4.4 Computational Results

The main goal of our computational study is to demonstrate that the proposed Benders decomposition algorithm – referred to as **BDS** in the rest – has a great computational performance both in absolute and relative terms. We solve instances across a broad range of  $(n, m)$  combinations with both short and long processing times and investigate the effectiveness of our algorithm in order to establish its absolute performance. Very large instances of both problems are within the reach of our algorithm. It turns out that **BDS** scales very well as instances with up to 1000 jobs and 6 machines are either solved to optimality within the time limit of one hour or very high-quality feasible solutions are obtained at termination. For  $n = 1000$ , the optimal solution is attained within 15.2 seconds for all instances for any  $m$ . Furthermore, to argue that **BDS** is the best exact algorithm for *Rm-UCDD* developed to date, we benchmark it against **CPX**, where the monolithic CQIP formulation (**CQ – U**), presented in Section 4.3, is solved directly by invoking CPLEX. As pointed out in Sections 4.2.1 and 4.3, **CPX** represents the current state-of-the-art for the exact methods designed for *Rm-UCDD*. The results reveal that compared to **CPX**, **BDS** either determines the optimal solution in considerably shorter time or it identifies an incumbent of substantially higher quality at the time limit. The details of our analyses are presented next.

Our instance generation follows suit with that of [Plateau and Rios-Solis \(2010\)](#) who evaluated (**CQ – U**) empirically. For each job  $j \in \{1, \dots, n\}$ , the processing time  $p_{jk}$  on machine  $k \in \{1, \dots, m\}$  and the unit earliness and tardiness penalties,  $\epsilon_j$  and  $\pi_j$ , are drawn from the discrete uniform distribution  $U[1, 20]$ . The unrestrictive common due date is set to  $\left\lceil \frac{\sum_j \max_k(p_{jk})}{m} \right\rceil + \max_{j,k}(p_{jk})$ . We create 10 instances for each combination of  $n \in \{10, 30, 50, 60, 80, 100, 400, 1000\}$  and  $m \in \{2, 4, 6\}$ . In this setup, the ratio  $\frac{n}{m}$  varies between 1.66 and 500 which allows us to explore the sensitivity of **BDS** to this parameter. In an effort to verify the robustness of **BDS** with respect to the range of the processing times, we repeat the same generation scheme except that the processing times are drawn from the discrete uniform distribution  $U[1, 100]$  – i.e.,  $p_{\max} = 100$  – which brings the total number of instances solved in this study to 480.

The computational results are obtained on a workstation with two 2.30GHz

Intel Xeon E5-2630 processors with Hyper-Threading enabled and 64 GB of memory running on Windows 8.1. **BDS** is implemented in C++ using the Concert Technology component library of IBM ILOG CPLEX 12.6. Note that in the presence of a control callback – such as the lazy constraint callback used in **BDS** – CPLEX switches off dynamic search feature, operates under deterministic parallel search mode, and apply a traditional branch-and-cut strategy with a single thread (IBM ILOG CPLEX, 2013). Therefore, to exploit parallelism and promote simultaneous cut generation, we set the `ParallelMode` switch to “opportunistic” parallel search mode and allowed CPLEX to use all available threads – i.e., the `Threads` parameter is set to 24. Furthermore, based on the positive previous experience of the authors in Chapters 2 and 3 the `MIPEmphasis` switch, which “controls the trade-offs between speed, feasibility, optimality, and moving bounds in MIP,” takes on the value four in order to urge CPLEX to find high-quality hidden feasible solutions. **CPX** calls CPLEX to solve (CQ – U) with the default parameter settings, except that `Threads=24`, `ParallelMode=Opportunistic`, and `MIPEmphasis=4` for a fair comparison with **BDS**. In both methods, CPLEX terminates the optimization if the relative optimality gap drops below  $\text{EpGap}=10^{-3}=0.1\%$ , or the working memory exceeds `WorkMem=5120=5 GB`, or the time expended reaches `TiLim=3600` seconds. More details on these parameters are available in (IBM ILOG CPLEX, 2013).

Table 4.1 consists of 24 rows, one for each possible combination of  $m$  and  $n$  listed in the first two columns. Each figure under the heading “Time” represents the average solution time statistic over 10 instances. The columns under “%Gap” present the average optimality gaps retrieved from CPLEX for the instances that are not solved to optimality within the time limit. The number of such instances is given next to the corresponding optimality gap result inside the parentheses. Note that CPLEX uses the formula  $\frac{|best\_bound - best\_integer|}{10^{-10} + |best\_integer|}$  for computing the optimality gap of an instance (IBM ILOG CPLEX, 2013), where *best\_bound* is the largest available lower bound and *best\_integer* is the objective value of the incumbent at termination. A color formatting scheme is applied separately to both performance measures, “%Gap” and “Time”, so that the values of a performance indicator ranging from better to worse are indicated with shades of red changing from light to dark. The results for instances with relatively short processing times are reported in the left half of the table in Columns 3–6 under the heading “ $p_{\max} = 20$ .” The remaining

columns depict the performance measures for the corresponding instances with  $p_{\max} = 100$ .

**Table 4.1** Average optimality gap and solution time results for *Rm-UCDD*

$m$	$n$	$p_{\max} = 20$				$p_{\max} = 100$			
		Time		%Gap <sup>†</sup>		Time		%Gap <sup>†</sup>	
		BDS	CPX	BDS	CPX	BDS	CPX	BDS	CPX
2	10	0.23	0.20			0.24	0.26		
	30	0.35	2.55			0.40	2.76		
	50	0.50	3.42			0.70	6.79		
	60	0.67	11.06			0.67	4.68		
	80	0.78	9.62			0.97	8.73		
	100	0.96	40.92			0.83	62.38		
	400	0.94	3.61			0.97	5.41		
	1000	2.06	19.42			2.23	21.68		
4	10	0.57	0.51			0.58	0.53		
	30	7.12	487		1.95 (1)	7.01	197		
	50	20.49	2560		2.49 (7)	58.82	3166		2.01 (8)
	60	59.01	2296		2.24 (6)	157	2542		2.24 (7)
	80	178	2889		1.69 (8)	327	2596		3.27 (6)
	100	363	2717		1.67 (7)	1114	2897	0.23 (1)	1.33 (8)
	400	4.93	2529		0.65 (7)	7.30	3452		0.38 (9)
	1000	5.37	2957		0.15 (8)	5.55	2174		0.15 (6)
6	10	5.09	7.42			2.60	12.74		
	30	623	3244		43.18 (9)	1483	3600	3.58 (2)	38.89 (10)
	50	2874	3600	1.56 (7)	16.22 (10)	3054	3600	2.36 (7)	16.49 (10)
	60	3600	2923	1.42 (10)	13.59 (8)	3512	3600	2.69 (9)	12.89 (10)
	80	3600	3600	1.26 (10)	10.96 (10)	3601	3548	1.92 (10)	12.72 (9)
	100	3600	3250	0.98 (10)	8.14 (9)	3601	3600	1.40 (10)	8.93 (10)
	400	1335	2262	0.13 (3)	2.24 (6)	3609	3304	0.14 (10)	1.19 (9)
	1000	11.85	2535		0.49 (7)	13.94	3528		0.42 (9)

<sup>†</sup>: The number of instances that are not solved to optimality within the time limit is given in parentheses and the percentage gap figures represent the averages over such instances.

The results in Table 4.1 underline that **BDS** provides provably optimal solutions for the majority of the instances well within the time limit of one hour. More specifically, **BDS** solves 391 out of a total of 480 instances to optimality in 122.8 seconds on average with a maximum solution time of 3445 seconds. Furthermore, the median solution time is 2.2 seconds and the solution time is less than 300 seconds for 91% of such instances (356 out of 391). In contrast, **CPX** attains only 266 optimal solutions in 106.2 seconds on average with a maximum of 3080 sec-

onds. Moreover, even though **CPX** is also able to obtain the optimal solutions for 94% of these instances (249 out of 266) within 300 seconds, the median time is 5.0 seconds. The average and maximum gaps of **BDS** for those 89 instances that could not be solved to optimality within the specified time limit are just 1.5% and 5.7%, respectively. The corresponding figures for **CPX** are 8.8% and 74.4% over 214 instances. Moreover, **BDS** reports less than 3% optimality gap for 91% of such instances (81 out of 89). Whereas, the corresponding number for **CPX** is just 53% (113 out of 214).

The differences between **BDS** and **CPX** become more apparent if we separate out the groups of instances solved to optimality by both methods and those not solved to optimality by either method within the time limit. On 207 of the 259 instances in the earlier group, **BDS** outpaces **CPX** by an average, median, and maximum factor of 18.5, 7.4, and 535, respectively, computed from the ratios of the solution times of **CPX** to those of **BDS**. On one instance the solution times are identical, and on the remaining 51 instances **CPX** is on average 4.2 times faster, where the corresponding median and maximum are 1.9 and 36, respectively. In the second group of 82 instances, **BDS** attains a smaller optimality gap at termination for 81 instances. The difference in the optimality gaps is on average 11.2 percentage points and reaches a maximum of 63.6. Whereas, **CPX** yields a smaller terminal gap on just one instance and the difference is 1.4 percentage points. In addition, note that there are only 7 instances for which **BDS** is only able to provide an incumbent at the time limit while **CPX** solves these instances optimally. The average and maximum gaps of **BDS** for those 7 instances are 0.8% and 2.1%, respectively. In comparison, **BDS** supplies optimal solutions for 132 instances that remain unsolved at the time limit by **CPX** with an average (& maximum) optimality gap of 6.5% (& 74.4%). To conclude, we stress that **BDS** is clearly the exact algorithm of choice for *Rm-UCDD* because it either delivers an optimal solution substantially faster or provides an incumbent with a much smaller optimality gap at termination.

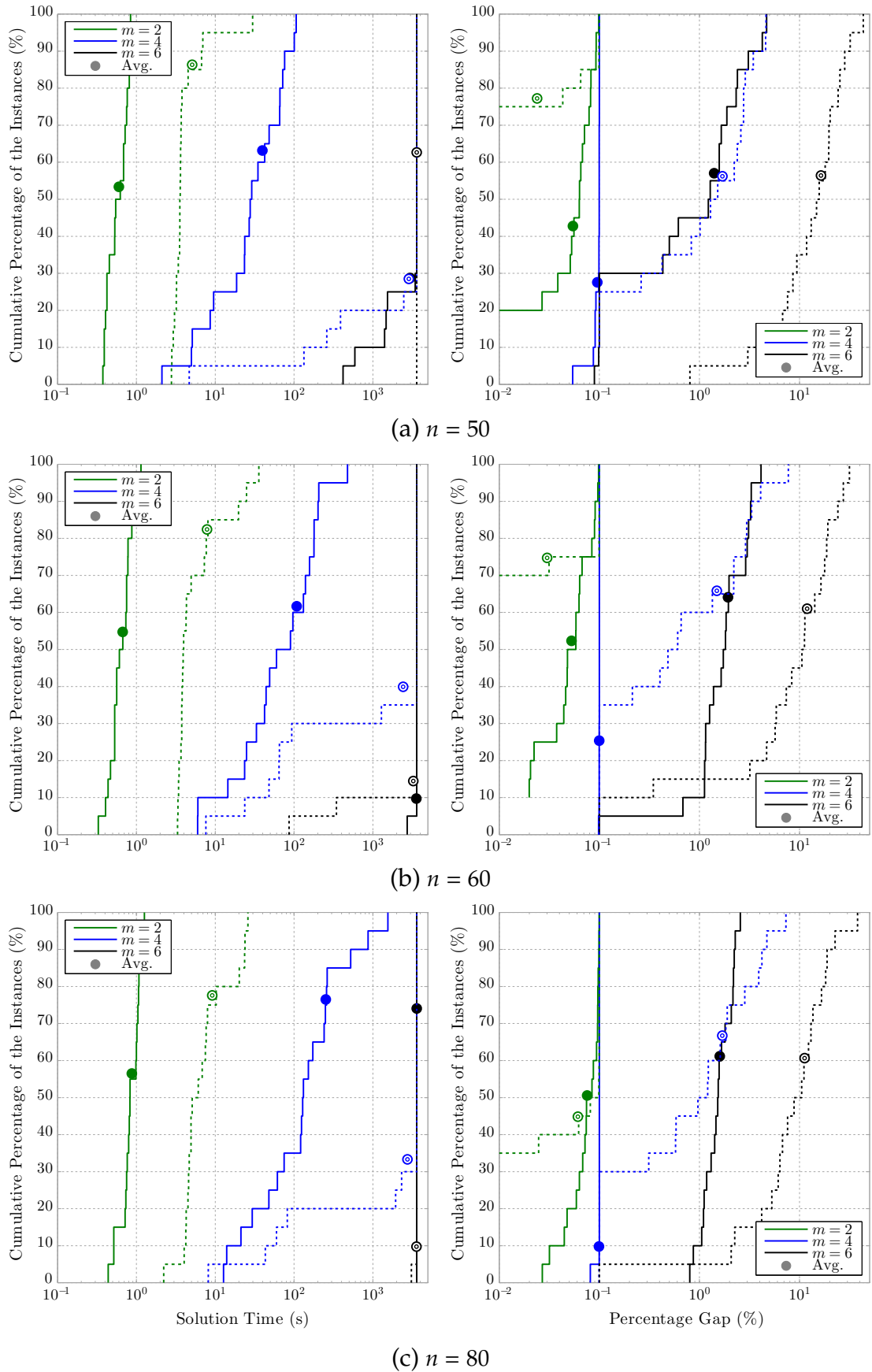
Table 4.1 attests to the solid performance of **BDS** regardless of the range of the processing times. The performance indicators related to **BDS** for both  $p_{\max} = 20$  and  $p_{\max} = 100$  are very similar. We reckon that two factors are at play here. First, the magnitude of the processing times has no effect on the size of the master

problem and the number of assignments variables. Second, the logic-based cuts are analytically generated.

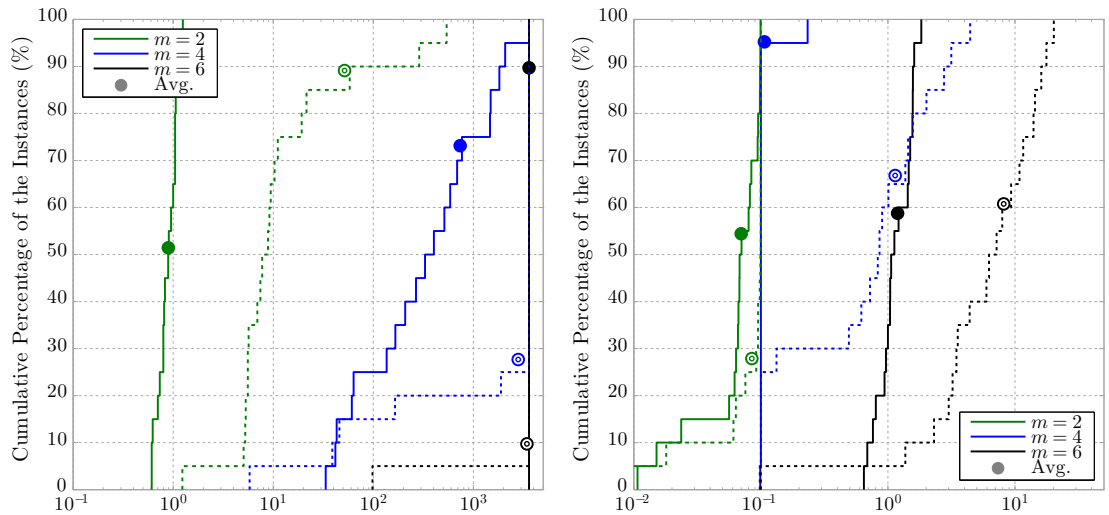
Next, we investigate how **BDS** and **CPX** scale with the number of jobs and machines. Contrary to the observation of [Ciré et al. \(2015\)](#) who state that the performance of LBB algorithms deteriorate if the average number of jobs assigned to each resource increases, for a fixed  $n$ , the solution times of **BDS** and **CPX** increase with  $m$ . That is, both methods favor larger  $\frac{n}{m}$  ratios. This may be regarded as a significant advantage since the more likely practical scenario is that  $n$  is significantly larger than  $m$ . Furthermore, observe that the solution times of **BDS** do not necessarily degrade with increasing  $n$  for a fixed  $m$ . Loosely speaking, the computational performance of **BDS** is determined by the number of machines. In contrast, the performance of **CPX** suffers from both higher  $n$  and  $m$  values.

Figures [4.1-4.2](#) further substantiate the robustness and scalability of **BDS** as an exact approach for  $Rm$ -UCDD. The empirical distributions of the solution times and the optimality gaps associated with both methods are depicted in these figures, where each curve is based on 20 instances – i.e., 10 instances with each possible  $p_{\max}$  values. The horizontal axes are in logarithmic scale to increase the readability of the graph. The median solution times and optimality gaps are associated with the 50% mark on the vertical axis, and the average gaps are explicitly indicated. Note that the shape of the optimality gap curves to the left of the  $10^{-1}\%$  mark do not bear any meaning because the relative optimality gap parameter of CPLEX is set to  $\text{EpGap} = 10^{-3} = 10^{-1}\%$ .

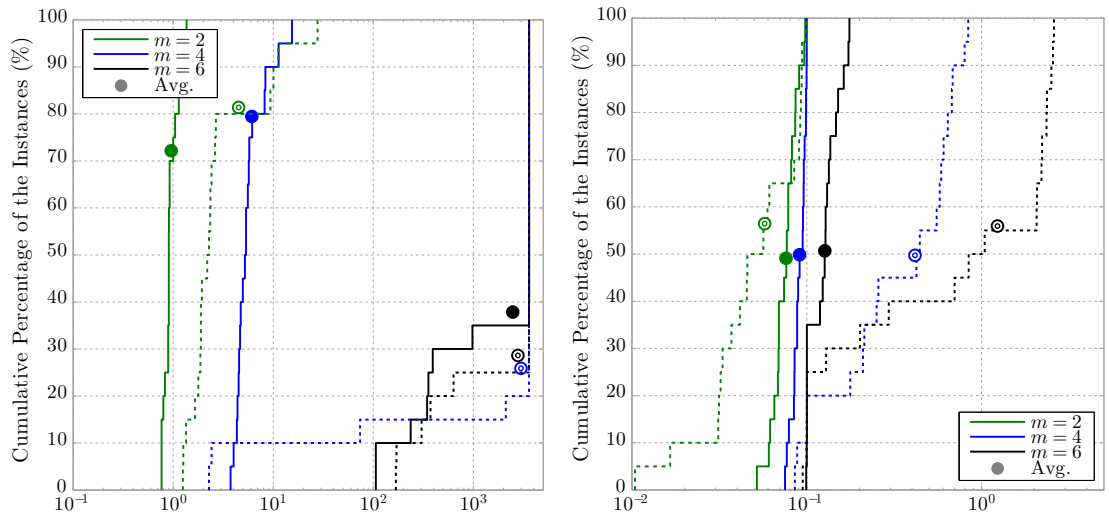
The relative insensitivity of **BDS** to  $n$  for a fixed  $m$  is also evident from Figure [4.1](#), where the curves for a fixed  $m$  are stacked on top of each other from Figure [4.1a](#) toward Figure [4.1c](#). The same phenomenon is still observable for  $m = 2$  on Figure [4.2](#). As stated previously, the number of machines is the main determinant of the solution time of **BDS**; the curves for a fixed  $n$  shift from left to right as  $m$  increases. Furthermore, we can also claim that **BDS** demonstrates a more consistent performance for these instances because the solution time curve for a given  $(n, m)$  combination rises relatively sharply and exhibits less variability across instances. Figure [4.1](#) confirms that the solution time performance of **BDS** is superior to that of **CPX** because the curves for **BDS** almost always lie to the left of the corresponding curves for **CPX**. A similar argument with respect to the



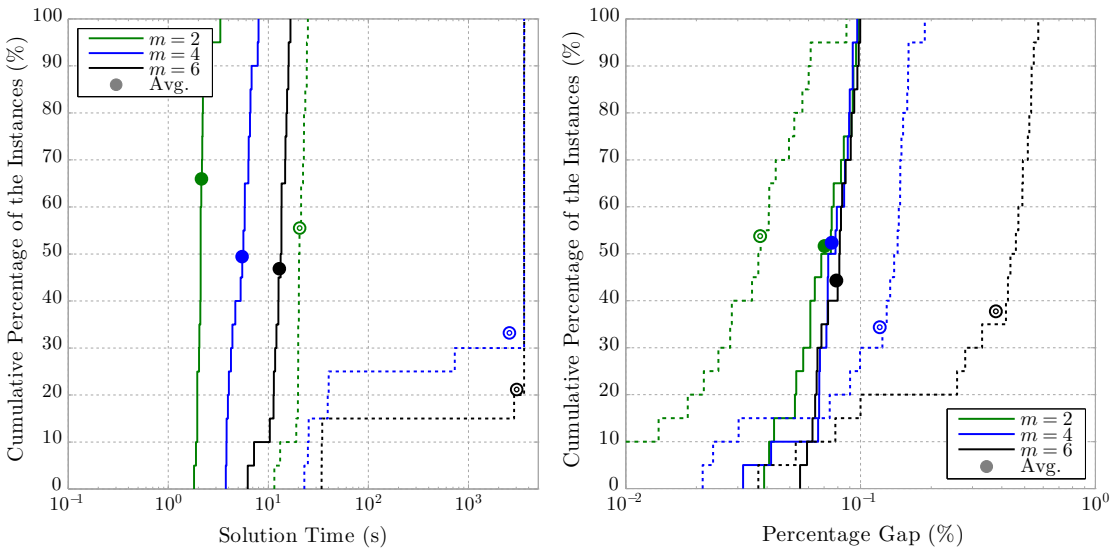
**Figure 4.1** The empirical distributions of the solution times and the optimality gaps of BDS (—) and CPX (---) for  $Rm$ -UCDD instances with 50, 60, and 80 jobs.



(a)  $n = 100$



(b)  $n = 400$



(c)  $n = 1000$

**Figure 4.2** The empirical distributions of the solution times and the optimality gaps of **BDS** (—) and **CPX** (---) for  $Rm$ -UCDD instances with 100, 400, and 1000 jobs.

solution times may be drawn from Figure 4.2 as well. The optimality gap curves of **BDS** clearly dominate those of **CPX**.

Finally, we note that the solution times of **BDS** are strongly correlated with both the number of Benders cuts generated and the number of nodes processed in the branch-and-cut search, as expected. Nevertheless, the duration of the cut generation procedure amounts to a small fraction of the total solution time and most of the cuts generated are in use at the end of the optimization. More specifically, the average share of the cut generation time within the total solution time is 0.5% with a corresponding median of 0.04%. The median and average percentages of the active Benders cuts for the final node problem in the search tree are both 88%. Overall, we may draw the conclusion that **BDS** is a scalable exact algorithm for *Rm-UCDD* and does either find the optimal solution considerably faster than the current state-of-the-art algorithm or it identifies considerably better incumbents at termination.



# CONCLUSION AND FUTURE RESEARCH

In Chapter 2, we developed a new preemptive relaxation for unrelated parallel machine scheduling problems with weighted tardiness and weighted earliness/tardiness objectives. The key property of this relaxation is that it provides us with a tight lower bound and a set of high-quality job partitions that forms the basis for the near-optimal non-preemptive solutions for the original problem. The relaxation itself is formulated as a difficult MIP problem, and a computationally effective Benders decomposition algorithm that can handle very large instances of this formulation is a primary contribution of this chapter. Our implementation employs state-of-the-art computational features, such as the *lazy constraint* callback of [IBM ILOG CPLEX \(2011\)](#) and a parallelization of the Benders subproblems via the Boost 1.51 library. Ultimately, we characterize our approach as a simple, non-parametric, and easy to implement mathematical programming based heuristic with a further distinguishing property that it can handle both a regular and a non-regular scheduling objective successfully with no additional customization. The results for *Rm-TWT* are outstanding. While those for *Rm-TWET* are not on a par, we reckon that they are of high quality.

In Chapter 3, we tackled the fundamental parallel machine scheduling problem *Rm-TWCT* which has been attacked by a variety of methodologies since the early 1970s. In a field dominated by custom B&B methods, approximation algorithms, and (meta-)heuristics, our approach makes elegant use of generic mathematical programming techniques. We refrain from a traditional and compact modeling

approach based on the job completion time variables and provide a new exact formulation of pseudo-polynomial size. Our formulation for  $Rm-TWCT$  is amenable to Benders decomposition, and we devise a computationally very effective algorithm that incorporates analytic solutions for the dual slave problems and a speedy cut strengthening procedure. The end product is a fast and scalable exact algorithm for  $Rm-TWCT$  which may even be employed as a subroutine in iterative decomposition-based algorithms developed for more complex shop scheduling problems.

In Chapter 4, we devised an LBB algorithm for unrelated parallel machine just-in-time scheduling with an unrestricted common due date. By extending the space of variables and analyzing the combinatorial structure of the subproblems, we demonstrated that it is possible to obtain a very strong bounding function for a scheduling problem with an irregular and additive performance measure. At the end of the day, the proposed exact algorithm is by far the best performing algorithm up to date for solving  $Rm-UCDD$  since it either solves the problem to optimality up to three orders of magnitude faster than the preceding state-of-the-art algorithm, or provides an incumbent with up to 75% smaller optimality gap at termination.

Initially, we also experimented with the identical parallel machine scheduling problems  $Pm//\sum_j \pi_j T_j$  and  $Pm//\sum_j \pi_j T_j + \epsilon_j E_j$ . However, the symmetry inherent in these problems results in many similar cuts and causes **(TR – A)-BDS** to choke. One of the items in our future research agenda is exploring ways of enhancing our algorithm to be able to handle the identical parallel machine environment. A further goal is to embed **(TR – A)-BDS** into an optimal algorithm for  $Rm-TWT$  and  $Rm-TWET$ . Note that the proposed preemptive relaxation can naturally handle branching decisions on the job to machine assignments.

Furthermore, the third and fourth chapters of this study demonstrate that moving away from the natural space of variables to an extended variable space may help attain better formulations and algorithms for machine scheduling problems. A research question worth investigating in the future is to explore scheduling problems in other domains that may benefit from similar techniques.

# Bibliography

- Alvarez-Valdes, R., Crespo, E., Tamarit, J., and Villa, F. (2012). Minimizing weighted earliness-tardiness on a single machine with a common due date using quadratic models. *TOP*, 20:754–767. (Cited on page 90.)
- Armentano, V. A. and Yamashita, D. S. (2000). Tabu search for scheduling on identical parallel machines to minimize mean tardiness. *J Intell Manuf*, 11(5):453–460. (Cited on pages 11, 12, and 15.)
- Azizoglu, M. and Kirca, O. (1998). Tardiness minimization on parallel machines. *Int J Prod Econ*, 55(2):163–168. (Cited on pages 8, 9, and 15.)
- Azizoglu, M. and Kirca, O. (1999a). On the minimization of total weighted flow time with identical and uniform parallel machines. *Eur J Oper Res*, 113(1):91–100. (Cited on pages 8 and 52.)
- Azizoglu, M. and Kirca, O. (1999b). Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. *IIE Trans*, 31(2):153–159. (Cited on pages 8, 9, 22, 52, and 65.)
- Baker, K. R. and Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: A review. *Oper Res*, 38(1):22–36. (Cited on pages 6, 16, 82, 83, and 87.)
- Barlatt, A. Y., Cohn, A. M., and Gusikhin, O. (2010). A hybridization of mathematical programming and dominance-driven enumeration for solving shift-selection and task-sequencing problems. *Comput Oper Res*, 37(7):1298–1307. (Cited on page 86.)
- Barnes, J. W. and Brennan, J. (1977). An improved algorithm for scheduling jobs on identical machines. *AIIE Transactions*, 9(1):25–31. (Cited on page 51.)

- Belouadah, H. and Potts, C. N. (1994). Scheduling identical parallel machines to minimize total weighted completion time. *Discrete Appl Math*, 48(3):201–218. (Cited on page 51.)
- Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numer Math*, 4(1):238–252. (Cited on pages 22, 26, 50, 57, and 85.)
- Benini, L., Bertozzi, D., Guerri, A., and Milano, M. (2005). Allocation and scheduling for MPSoCs via decomposition and no-good generation. In van Beek, P., editor, *Principles and Practice of Constraint Programming - CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 107–121. Springer Berlin Heidelberg. (Cited on page 86.)
- Berghman, L., Spieksma, F., and T'Kindt, V. (2014). Solving a time-indexed formulation by preprocessing and cutting planes. Available at SSRN: <http://dx.doi.org/10.2139/ssrn.2437371>. (Cited on page 54.)
- Beyranvand, M., Peyghami, M., and Ghatee, M. (2012). On the quadratic model for unrelated parallel machine scheduling problem with restrictive common due date. *Optim Lett*, 6(8):1897–1911. (Cited on page 84.)
- Billionnet, A., Elloumi, S., and Plateau, M.-C. (2009). Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method. *Discrete Appl Math*, 157(6):1185 – 1197. (Cited on page 84.)
- Biskup, D., Herrmann, J., and Gupta, J. N. (2008). Scheduling identical parallel machines to minimize total tardiness. *Int J Prod Econ*, 115(1):134–142. (Cited on pages 1, 11, and 15.)
- Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Weglarz, J. (2007). *Handbook on scheduling: from theory to applications*. Springer. (Cited on page 51.)
- Bockmayr, A. and Kasper, T. (1998). Branch and infer: A unifying framework for integer and finite domain constraint programming. *INFORMS J Comput*, 10(3):287–300. (Cited on page 85.)

- Bockmayr, A. and Pisaruk, N. (2003). Detecting infeasibility and generating cuts for MIP using CP. In *5th International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems - CPAIOR'03*, Montréal, Canada. <https://hal.inria.fr/inria-00107699>. (Cited on page 86.)
- Bruno, J., Coffman Jr, E. G., and Sethi, R. (1974). Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17(7):382–387. (Cited on page 49.)
- Bülbül, K., Kaminsky, P., and Yano, C. (2007). Preemption in single machine earliness/tardiness scheduling. *J Sched*, 10(4-5):271–292. (Cited on pages 7, 17, 19, 21, 55, and 59.)
- Bülbül, K. and Şen, H. (2015). An exact extended formulation for the unrelated parallel machine total weighted completion time problem. Submitted for publication. [http://www.optimization-online.org/DB\\_HTML/2014/08/4519.html](http://www.optimization-online.org/DB_HTML/2014/08/4519.html). (Cited on page 4.)
- Burkard, R., Dell'Amico, M., and Martello, S. (2009). *Assignment Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. (Cited on pages 60 and 61.)
- Cambazard, H., Hladik, P.-E., Déplanche, A.-M., Jussien, N., and Trinquet, Y. (2004). Decomposition and learning for a hard real time task allocation problem. In Wallace, M., editor, *Principles and Practice of Constraint Programming – CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pages 153–167. Springer Berlin Heidelberg. (Cited on page 86.)
- Chekuri, C. and Khanna, S. (2004). Approximation algorithms for minimizing average weighted completion time. In Leung, J. Y., editor, *Handbook of scheduling: algorithms, models, and performance analysis*. CRC Press, Boca Raton, FL, USA, 2004. (Cited on page 51.)
- Chen, Z.-L. and Lee, C.-Y. (2002). Parallel machine scheduling with a common due window. *Eur J Oper Res*, 136(3):512–527. (Cited on pages 10, 15, and 83.)

- Chen, Z.-L. and Powell, W. B. (1999a). A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. *Eur J Oper Res*, 116(1):220–232. (Cited on pages 10, 15, and 83.)
- Chen, Z.-L. and Powell, W. B. (1999b). Solving parallel machine scheduling problems by column generation. *INFORMS J Comput*, 11(1):78–94. (Cited on pages 8, 53, 54, 73, and 76.)
- Cheng, T. and Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *Eur J Oper Res*, 47(3):271–292. (Cited on pages 8, 16, and 51.)
- Ciré, A. A., Çoban, E., and Hooker, J. N. (2015). Logic-based benders decomposition for planning and scheduling: A computational analysis. In *Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS-15)*, pages 21–29. <http://www.cs.bgu.ac.il/~icaps15/workshops/ProceedingsCOPLAS2015.pdf>. (Cited on pages 86, 92, and 105.)
- Coban, E. and Hooker, J. N. (2013). Single-facility scheduling by logic-based benders decomposition. *Ann Oper Res*, 210(1):245–272. (Cited on pages 87 and 92.)
- Detienne, B., Dauzère-Pérès, S., and Yugma, C. (2011). Scheduling jobs on parallel machines to minimize a regular step total cost function. *J Sched*, 14:523–538. (Cited on page 1.)
- Dyer, M. and Wolsey, L. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Appl Math*, 26(2–3):255–270. (Cited on pages 54 and 62.)
- Elmaghraby, S. E. and Park, S. H. (1974). Scheduling jobs on a number of identical machines. *AIIE transactions*, 6(1):1–13. (Cited on page 51.)
- Fischetti, M., Salvagnin, D., and Zanette, A. (2010). A note on the selection of Benders' cuts. *Math Program*, 124(1-2):175–182. (Cited on pages 27 and 69.)
- Goemans, M. X., Queyranne, M., Schulz, A. S., Skutella, M., and Wang, Y. (2002). Single machine scheduling with release dates. *SIAM J Discrete Math*, 15(2):165–192. (Cited on page 62.)

- Graham, R., Lawler, E., Lenstra, J., and Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E. J. and Korte, B., editors, *Discrete Optimization II*, volume 5 of *Ann Discrete Math*, pages 287–326. Elsevier. (Cited on pages 6, 49, and 81.)
- Hall, N. G. and Posner, M. E. (1991). Earliness-tardiness scheduling problems, i: Weighted deviation of completion times about a common due date. *Oper Res*, 39(5):836–846. (Cited on page 90.)
- Harjunkski, I. and Grossmann, I. E. (2001). A decomposition approach for the scheduling of a steel plant production. *Comput Chem Eng*, 25(11):1647–1660. (Cited on page 86.)
- Harjunkski, I. and Grossmann, I. E. (2002). Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Comput Chem Eng*, 26(11):1533–1552. (Cited on page 86.)
- Hooker, J. N. (2000). *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. John Wiley & Sons, Inc., New York. (Cited on pages 85 and 89.)
- Hooker, J. N. (2004). A hybrid method for planning and scheduling. In Wallace, M., editor, *Principles and Practice of Constraint Programming – CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pages 305–316. Springer Berlin Heidelberg. (Cited on pages 86 and 92.)
- Hooker, J. N. (2005a). A hybrid method for the planning and scheduling. *Constraints*, 10(4):385–401. (Cited on pages 86 and 92.)
- Hooker, J. N. (2005b). Planning and scheduling to minimize tardiness. In van Beek, P., editor, *Principles and Practice of Constraint Programming - CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 314–327. Springer Berlin Heidelberg. (Cited on pages 86 and 92.)
- Hooker, J. N. (2006). An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11(2-3):139–157. (Cited on pages 86 and 92.)

- Hooker, J. N. (2007a). *Integrated Methods for Optimization*. International series in operations research & management science. Springer-Verlag New York, Inc., Secaucus, NJ, USA. (Cited on page 85.)
- Hooker, J. N. (2007b). Planning and scheduling by logic-based benders decomposition. *Oper Res*, 55(3):588–602. (Cited on pages 86 and 92.)
- Hooker, J. N. and Ottosson, G. (2003). Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60. (Cited on pages 81, 85, 88, and 92.)
- IBMILOG CPLEX (2011). IBM ILOG CPLEX Optimization Studio 12.4 Information Center. <http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/index.jsp>. Last viewed on 04/24/2013. (Cited on pages 7, 31, 34, and 109.)
- IBMILOG CPLEX (2012). IBM ILOG CPLEX Optimization Studio 12.5 Information Center. <http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r5/index.jsp>. Last viewed on 08/04/2014. (Cited on pages 65, 73, and 74.)
- IBMILOG CPLEX (2013). IBM ILOG CPLEX Optimization Studio 12.6 Information Center. <http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r6/index.jsp>. Last viewed on 06/12/2015. (Cited on pages 100 and 102.)
- Jain, V. and Grossmann, I. E. (2001). Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS J Comput*, 13(4):258–276. (Cited on pages 85 and 86.)
- Jouglet, A. and Savourey, D. (2011). Dominance rules for the parallel machine total weighted tardiness scheduling problem with release dates. *Comput Oper Res*, 38(9):1259–1266. (Cited on pages 10 and 15.)
- Kanet, J. J. (1981). Minimizing the average deviation of job completion times about a common due date. *Nav Res Log Quarterly*, 28(4):643–651. (Cited on page 83.)
- Kanet, J. J. and Sridharan, V. (2000). Scheduling with inserted idle time: Problem taxonomy and literature review. *Oper Res*, 48(1):99–110. (Cited on pages 6, 16, 82, and 87.)



- Kedad-Sidhoum, S., Solis, Y. R., and Sourd, F. (2008). Lower bounds for the earliness–tardiness scheduling problem on parallel machines with distinct due dates. *Eur J Oper Res*, 189(3):1305–1316. (Cited on pages [12](#), [13](#), [14](#), [15](#), [18](#), [21](#), [33](#), [35](#), and [56](#).)
- Koulamas, C. (1997). Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem. *Nav Res Log*, 44(1):109–125. (Cited on pages [9](#), [11](#), and [15](#).)
- Lauff, V. and Werner, F. (2004). Scheduling with common due date, earliness and tardiness penalties for multimachine problems: A survey. *Math Comput Model*, 40(5):637–655. (Cited on pages [16](#) and [83](#).)
- Lawler, E. L. and Moore, J. M. (1969). A functional equation and its application to resource allocation and sequencing problems. *Manage Sci*, 16(1):77–84. (Cited on page [51](#).)
- Lee, C.-Y. and Uzsoy, R. (1992). A new dynamic programming algorithm for the parallel machines total weighted completion time problem. *Oper Res Lett*, 11(2):73–75. (Cited on page [51](#).)
- Lenstra, J., Rinnooy Kan, A., and Brucker, P. (1977). Complexity of machine scheduling problems. *Ann Discrete Math*, 1:343–362. (Cited on page [6](#).)
- Li, K. and Yang, S.-l. (2009). Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied mathematical modelling*, 33(4):2145–2158. (Cited on pages [51](#) and [54](#).)
- Liaw, C.-F., Lin, Y.-K., Cheng, C.-Y., and Chen, M. (2003). Scheduling unrelated parallel machines to minimize total weighted tardiness. *Comput Oper Res*, 30(12):1777–1789. (Cited on pages [9](#), [15](#), and [33](#).)
- Lin, Y., Pfund, M., and Fowler, J. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Comput Oper Res*, 38(6):901–916. (Cited on pages [12](#), [15](#), [33](#), and [50](#).)

- Luh, P. B., Hoitomt, D. J., Max, E., and Pattipati, K. R. (1990). Schedule generation and reconfiguration for parallel machines. *IEEE T Robot Autom*, 6(6):687–696. (Cited on pages [11](#), [12](#), and [15](#).)
- Magnanti, T. L. and Wong, R. T. (1981). Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Oper Res*, 29(3):464–484. (Cited on pages [27](#) and [69](#).)
- Maravelias, C. T. and Grossmann, I. E. (2004). A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Comput Chem Eng*, 28(10):1921–1949. (Cited on page [86](#).)
- Mason, S. J., Jin, S., and Jampani, J. (2009). A moving block heuristic to minimise earliness and tardiness costs on parallel machines. *Int J Prod Res*, 47(19):5377–5390. (Cited on pages [12](#), [13](#), and [16](#).)
- M'Hallah, R. and Al-Khamis, T. (2012). Minimising total weighted earliness and tardiness on parallel machines using a hybrid heuristic. *Int J Prod Res*, 50(10):2639–2664. (Cited on pages [12](#), [13](#), and [16](#).)
- Mokotoff, E. (2001). Parallel machine scheduling problems: a survey. *Asia Pacific Journal of Operational Research*, 18(2):193–242. (Cited on page [51](#).)
- Mönch, L. (2008). Heuristics to minimize total weighted tardiness of jobs on unrelated parallel machines. In *2008 IEEE Inter Conf on Automation Science and Engineering*, pages 572–577. IEEE. (Cited on pages [12](#) and [15](#).)
- Nessah, R., Yalaoui, F., and Chu, C. (2008). A branch-and-bound algorithm to minimize total weighted completion time on identical parallel machines with job release dates. *Computers & Operations Research*, 35(4):1176–1190. (Cited on page [52](#).)
- Pan, Y. and Shi, L. (2007). On the equivalence of the max-min transportation lower bound and the time-indexed lower bound for single-machine scheduling problems. *Math Program*, 110(3):543–559. (Cited on pages [7](#), [19](#), [21](#), [55](#), and [58](#).)

- Pessoa, A., Uchoa, E., Aragão, M. P., and Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Math Program Comput*, 2(3-4):259–290. (Cited on pages 10 and 15.)
- Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer, 3rd edition. (Cited on pages 1, 8, and 51.)
- Plateau, M.-C. and Rios-Solis, Y. A. (2010). Optimal solutions for unrelated parallel machines scheduling problems using convex quadratic reformulations. *Eur J Oper Res*, 201(3):729–736. (Cited on pages 15, 53, 55, 73, 81, 84, 88, 90, and 101.)
- Posner, M. E. (1985). Minimizing weighted completion times with deadlines. *Operations Research*, 33(3):562–574. (Cited on page 62.)
- Potts, C. and van Wassenhove, L. (1982). A decomposition algorithm for the single machine total tardiness problem. *Oper Res Lett*, 1(5):177–181. (Cited on page 33.)
- Rasmussen, R. V. and Trick, M. A. (2007). A benders approach for the constrained minimum break problem. *Eur J Oper Res*, 177(1):198–213. (Cited on page 86.)
- Rios-Solis, Y. A. and Sourd, F. (2008). Exponential neighborhood search for a parallel machine scheduling problem. *Comput Oper Res*, 35(5):1697–1712. (Cited on pages 15 and 83.)
- Rodriguez, F., Blum, C., García-Martínez, C., and Lozano, M. (2012). GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times. *Ann Oper Res*, 201(1):383–401. (Cited on page 50.)
- Rodriguez, F. J., Lozano, M., Blum, C., and García-Martínez, C. (2013). An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. *Comput Oper Res*, 40(7):1829–1841. (Cited on pages 50, 51, and 55.)
- Rubin, P. (2011). Benders decomposition then and now. <http://orinanobworld.blogspot.com/2011/10/benders-decomposition-then-and-now.html>. Last viewed on 04/24/2013. (Cited on pages 31, 65, and 100.)

- Sadykov, R. and Wolsey, L. A. (2006). Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. *INFORMS J Comput*, 18(2):209–217. (Cited on page 86.)
- Sarin, S. C., Ahn, S., and Bishop, A. B. (1988). An improved branching scheme for the branch and bound procedure of scheduling  $n$  jobs on  $m$  parallel machines to minimize total weighted flowtime. *International Journal of Production Research*, 26(7):1183–1191. (Cited on page 51.)
- Şen, H. and Bülbül, K. (2012). A simple, fast, and effective heuristic for the single-machine total weighted tardiness problem. In Demeulemeester, E. and Herroelen, W., editors, *Proceedings of the 13th Inter. Conf. on Project Management and Scheduling (PMS 2012)*, pages 282–286, Leuven, Belgium. (Cited on pages 7, 17, 19, and 55.)
- Şen, H. and Bülbül, K. (2015a). Logic based benders decomposition for unrelated parallel machine weighted earliness/tardiness scheduling. Manuscript in preparation. (Cited on page 4.)
- Şen, H. and Bülbül, K. (2015b). A strong preemptive relaxation for weighted tardiness and earliness/tardiness problems on unrelated parallel machines. *INFORMS J Comput*, 27(1):135–150. (Cited on pages 4 and 87.)
- Sen, T., Sulek, J. M., and Dileepan, P. (2003). Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. *Int J Prod Econ*, 83(1):1 – 12. (Cited on pages 10 and 16.)
- Shim, S.-O. and Kim, Y.-D. (2007a). Minimizing total tardiness in an unrelated parallel-machine scheduling problem. *J Oper Res Soc*, 58(3):346–354. (Cited on pages 1, 2, 6, 9, 15, and 33.)
- Shim, S.-O. and Kim, Y.-D. (2007b). Scheduling on parallel identical machines to minimize total tardiness. *Eur J Oper Res*, 177(1):135–146. (Cited on pages 9 and 15.)
- Skutella, M. (2001). Convex quadratic and semidefinite programming relaxations in scheduling. *J ACM*, 48(2):206–242. (Cited on pages 53, 55, and 84.)

- Smith, W. E. (1956). Various optimizers for single-stage production. *Nav Res Log*, 3(1-2):59–66. (Cited on pages 49 and 90.)
- Souayah, N., Kacem, I., Haouari, M., and Chu, C. (2009). Scheduling on parallel identical machines to minimise the total weighted tardiness. *Inter J Adv Oper Manage*, 1(1):30–69. (Cited on pages 9, 10, and 15.)
- Soukhal, A. and Toung, N. (2012). Just-in-time scheduling with equal-size jobs. In Ríos-Mercado, R. Z. and Ríos-Solís, Y. A., editors, *Just-in-Time Systems*, volume 60 of *Springer Optimization and Its Applications*, pages 107–145. Springer New York. (Cited on page 84.)
- Sourd, F. and Kedad-Sidhoum, S. (2003). The one-machine problem with earliness and tardiness penalties. *J Sched*, 6(6):533–549. (Cited on pages 7, 17, 19, 21, and 55.)
- Tanaka, S. and Araki, M. (2008). A branch-and-bound algorithm with Lagrangian relaxation to minimize total tardiness on identical parallel machines. *Int J Prod Econ*, 113(1):446–458. (Cited on pages 10, 11, 12, 13, and 15.)
- Tanaka, S. and Fujikuma, S. (2012). A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. *J Sched*, 15:347–361. (Cited on pages 7, 22, 34, and 40.)
- Tanaka, S., Fujikuma, S., and Araki, M. (2009). An exact algorithm for single-machine scheduling without machine idle time. *J Sched*, 12:575–593. (Cited on pages 7, 22, 34, and 40.)
- Thorsteinsson, E. (2001). Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In Walsh, T., editor, *Principles and Practice of Constraint Programming - CP 2001*, volume 2239 of *Lecture Notes in Computer Science*, pages 16–30. Springer Berlin Heidelberg. (Cited on pages 85 and 86.)
- Timpe, C. (2002). Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum*, 24(4):431–448. (Cited on page 86.)

- Unlu, Y. and Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Comput Ind Eng*, 58(4):785–800. (Cited on page 54.)
- Üster, H. and Agrahari, H. (2011). A Benders decomposition approach for a distribution network design problem with consolidation and capacity considerations. *Oper Res Lett*, 39(2):138–143. (Cited on page 27.)
- van den Akker, J. M., Hoogeveen, J. A., and van de Velde, S. L. (1999). Parallel Machine Scheduling by Column Generation. *Oper Res*, 47(6):862–872. (Cited on pages 7, 8, 50, 53, 54, 73, and 76.)
- Van Roy, T. J. (1986). A cross decomposition algorithm for capacitated facility location. *Oper Res*, 34(1):145–163. (Cited on page 27.)
- Vredeveld, T. and Hurkens, C. (2002). Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. *INFORMS Journal on Computing*, 14(2):175–189. (Cited on page 50.)
- Webster, S. (1997). The complexity of scheduling job families about a common due date. *Oper Res Lett*, 20(2):65–74. (Cited on page 81.)
- Wentges, P. (1996). Accelerating Benders' decomposition for the capacitated facility location problem. *Math Method Oper Res*, 44(2):267–290. (Cited on page 27.)
- Yalaoui, F. and Chu, C. (2002). Parallel machine scheduling to minimize total tardiness. *Int J Prod Econ*, 76(3):265–279. (Cited on pages 9, 11, and 15.)
- Yalaoui, F. and Chu, C. (2006). New exact method to solve the  $Pm/r_j/\sum_j C_j$  schedule problem. *International Journal of Production Economics*, 100(1):168–179. (Cited on page 52.)
- Zhou, H., Li, Z., and Wu, X. (2007). Scheduling Unrelated Parallel Machine to Minimize Total Weighted Tardiness Using Ant Colony Optimization. In *2007 IEEE Inter Conf on Automation and Logistics*, pages 132–136, Jinan. IEEE. (Cited on pages 12 and 15.)

# Curriculum Vitae

## Halil Şen

halilsen@sabanciuniv.edu ✉

[myweb.sabanciuniv.edu/halilsen](http://myweb.sabanciuniv.edu/halilsen) 🌐

### Education

- SABANCI UNIVERSITY İSTANBUL, TR  
*Ph.D., Industrial Engineering* 08/2015  
• Dissertation: *Cut Generation Based Algorithms for Unrelated Parallel Machine Scheduling Problems* Advisor: Kerem Bülbül
- SABANCI UNIVERSITY İSTANBUL, TR  
*M.Sc., Industrial Engineering* 08/2010  
• Thesis: *A Simple, Fast, and Effective Heuristic for the Single-Machine Total Weighted Tardiness Problem* Advisor: Kerem Bülbül
- YILDIZ TECHNICAL UNIVERSITY İSTANBUL, TR  
*B.Sc., Industrial Engineering* 06/2008

### Academic Experience

- RESEARCH AND TEACHING ASSISTANT 09/2008–08/2015  
*Sabancı University* İSTANBUL, TR
- COLLABORATION 02/2013–06/2015  
*Pierre and Marie Curie University, LIP6, Safia Kedad-Sidhoum* PARIS, FR  
• Research: *Single-Machine Earliness-Tardiness Scheduling with Periods of Machine Unavailability*
- VISITING RESEARCHER 02–06/2014  
*The Ohio State University, ISE, Simge Küçükyavuz* COLUMBUS, OH  
• Research: *Chance-Constrained Two-Stage Mean-Risk Stochastic Programming*

## Fellowship & Awards

- Third prize, Best Student Paper Competition, (PMS 2012), Leuven, BE 2012
- TÜBİTAK M.Sc. and Ph.D. Scholarships 2008–present
- Sabancı University M.Sc. and Ph.D. Fellowships 2008–present
- Graduated from Yıldız Technical University as a student with honors 2008
- Turkish Prime Ministry Scholarship 2004–2008

## Published/Submitted Journal Papers

Şen, Halil and Bülbül, K. (2015). A Strong Preemptive Relaxation for Weighted Tardiness and Earliness/Tardiness Problems on Unrelated Parallel Machines. *INFORMS Journal on Computing*, 27(1), 135–150. <http://dx.doi.org/10.1287/ijoc.2014.0615>

Bülbül, Kerem and Şen, H. (2014). An Exact Extended Formulation for the Unrelated Parallel Machine Total Weighted Completion Time Problem. Manuscript submitted for publication.

[www.optimization-online.org/DB\\_HTML/2014/08/4519.html](http://www.optimization-online.org/DB_HTML/2014/08/4519.html)

## Proceedings

Şen, Halil and Bülbül, K. (2012). A Simple, Fast, and Effective Heuristic for the Single Machine Total Weighted Tardiness Problem. In Demeulemeester, E. and Herroelen, W. (Eds.), *Proceedings of the 13th Inter. Conf. on Project Management and Scheduling*, 282–286.

[www.econ.kuleuven.be/eng/tew/academic/prodbel/PMS2012/proceedings.pdf](http://www.econ.kuleuven.be/eng/tew/academic/prodbel/PMS2012/proceedings.pdf)

## Manuscripts in Preparation

- *Chance-Constrained Two-Stage Mean-Risk Stochastic Programming*. Joint work with K. Bülbül, S. Küçükyavuz, and N. Noyan.
- *Single-Machine Earliness-Tardiness Scheduling with Periods of Machine Unavailability*. Joint work with K. Bülbül and S. Kedad-Sidhoum.
- *Logic-Based Benders Decomposition for Unrelated Parallel Machine Weighted Earliness/Tardiness Scheduling*. Joint work with K. Bülbül.



*A Strong Preemptive Relaxation for Minsum Scheduling Problems on Unrelated Parallel Machines*

Joint work with K. Bülbül.

- University of Bordeaux, Seminar Series of ReAlOpt, Bordeaux, FR, 2015
- The Université Catholique de Louvain, CORE Seminar Series, Louvain-la-Neuve, BE, 2015
- Sabancı University, IE-OPIM Joint Graduate Seminar Series, İstanbul, TR, 2014 <sup>†</sup>
- University of Southern California, Los Angeles, CA, 2014 <sup>†</sup>

*Single-Machine Earliness-Tardiness Scheduling with Periods of Machine Unavailability*

Joint work with K. Bülbül and S. Kedad-Sidhoum.

- 20th Conference of the International Federation of OR Societies (IFORS 2014), Barcelona, ES, 2014 (presented by S. Kedad-Sidhoum)

*A Strong Preemptive Relaxation for Weighted Tardiness and Earliness/Tardiness Problems on*

*Unrelated Parallel Machines* Joint work with K. Bülbül.

- OR/IE Doctoral Students Colloquium, Middle East Technical University, Ankara, TR, 2014
- The Ohio State University, ISE Seminar Series, Columbus, OH, 2014 <sup>†</sup>
- 33rd National Conference on OR and Industrial Engineering (YAEM) – joint with the International IIE Conference, İstanbul, TR, 2013
- XXVI EURO – INFORMS Joint International Conference, Rome, IT, 2013 <sup>†</sup>
- Joint 2nd Workshop of the Turkish and Israeli OR Societies (WITOR2), Tel Aviv, IL, 2013 <sup>†</sup>

*A Preemption-Based Heuristic for the Single-Machine Generalized Total Weighted Tardiness*

Joint work with K. Bülbül.

- 13th Inter. Conf. on Project Management and Scheduling (PMS 2012), Leuven, BE, 2012
- 32nd National Conference on OR and Industrial Engineering (YAEM), İstanbul, TR, 2012

*A Simple, Fast, and Effective Heuristic for the Single-Machine Total Weighted Tardiness*

- Turkish Naval Academy, İstanbul, TR, 2012
- 24th European Conference on Operational Research (EURO XXIV), Lisbon, PT, 2010
- 30th National Conference on OR and Industrial Engineering (YAEM), İstanbul, TR, 2010