2017-04-17

Single Machine Preemptive Scheduling for Tardiness Related Objectives: Exact Methods and Heuristics

Fernando Jaramillo
University of Miami, fernando.jaramillo.a@gmail.com

UNIVERSITY OF MIAMI

SINGLE MACHINE PREEMPTIVE SCHEDULING FOR TARDINESS RELATED OBJECTIVES: EXACT METHODS AND HEURISTICS

By

Fernando Jaramillo A.

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

May 2017

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

SINGLE MACHINE PREEMPTIVE SCHEDULING FOR TARDINESS RELATED OBJECTIVES: EXACT METHODS AND HEURISTICS

Fernando Jaramillo A.

Approved:

Murat Erkoc, Ph.D. Associate Professor of Industrial Engineering	Nazrul Shaikh, Ph.D. Assistant Professor of Industrial Engineering
Shihab S. Asfour, Ph.D. Professor and Associate Dean of Industrial Engineering	Guillermo Prado, Ph.D. Dean of the Graduate School
Harihara P. Natarajan, Ph.D. Associate Professor of Management	

JARAMILLO A., FERNANDO (Ph.D., Industrial Engineering)

Single Machine Preemptive Scheduling for Tardiness Related Objectives: Exact Methods and Heuristics (May 2017)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Murat Erkoc.
No. of pages in text. (148)

We explore a set of diverse tardiness related problems for single machine preemptive scheduling in their most general forms, and for each one of these problems we compare the performance of different solution approaches. We start by comparing the performance of two proposed heuristics versus the conventional modeling approach for the solution of the Total Weighted Tardiness problem with and without overtime capacity allocation. We continue the discussion focusing only on exact methods, and we do this by comparing the performance between the conventional model and an advanced model that borrows from the aggregate planning paradigm. We conclude our discussion by comparing the conventional and advanced modeling approaches under a set of diverse tardiness related problems including: Total Weighed Tardiness, Total Weighted Completion, Total Weighted Earliness and Tardiness and Total Weighted Number of Tardy Jobs. Via numerical experimentation and analytical methods we show that the the proposed heuristics as well as the advanced modeling approaches are more efficient at obtaining good quality solutions than their corresponding conventional models. For larger size problems, the advanced modeling approach and the heuristics represent more efficient tools for generating optimal or near-optimal schedules to a variety of tardiness related problems.

To my parents who provided me with the encouragement

to carry this work to its end

Acknowledgements

Special thanks to my advisor Dr. Murat Erkoc for his support and guidance, and my committee members Dr. Shihab Asfour, Dr. Nazrul Shaikh and Dr. Harihara Natarajan for their valuable feedback on making this dissertation a more robust piece of academic research.

Fernando Jaramillo A.

University of Miami

May 2017

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

From a purely theoretical perspective, this document can be seen as a brief collection of methodologies for the optimization of single machine (single processor) schedules with the objective to minimize tardiness related measures; however, introducing from that perspective this work wouldn't be meaningful as it would lack the context and practical application that such methodologies have. With that in mind, our work is mainly motivated by the operations of a major aviation maintenance, repair and overhaul (MRO) company, and the proposed methodologies here described can be applied to the wide variety of services present in the MRO sector, and specifically to the regulated MRO sub-sector where maintenance and repair operations are overseen by regulating agencies such as the FAA, EPA or FDA among others. In what follows we will refer to this sub sector as R-MRO.

The R-MRO sector is very particular in the sense that its services are provided to customers with large capital investments on equipment, and service turnaround time is also directly linked to significant revenue losses during periods of equipment downtime. Within the R-MRO services there is a high financial sensitivity to time; this is, two service schedules that differ on a single day of delay in the repair of a commercial airplane, represent an average cost of 50,000.00 USD for the equipment

owner and thousands in tardiness penalties to the service provider not to mention less quantifiable costs such as good-will losses.

Considering the importance behind minimizing the losses and costs that make part of a tardiness related event in the R-MRO sector, it seems contradicting that in today's practice largely sub-optimal methods like the first-come-first-served (FCFS) rule represent the scheduling tool most commonly used; the reason for such a contradiction results from the complexity of the scheduling problems at hand which in the current state of optimization sciences lack of accurate and efficient tools that allow the generation of optimal or near optimal schedules in a reasonable amount of time.

Here we propose and study a set of methods that enhance the scheduling ability of a maintenance service provider beyond the FCFS rule; We engage in the discussion of approximation algorithms and exact methods (heuristics and linear programming models) tailored to the problem of optimizing the schedule of a single processor where the goal is to minimize a tardiness related measure. For all the tardiness related measures we discuss (also known as tardiness related problems) we assume that the processing environment can be accurately modeled by a single processor operating under finite capacity over a finite number of jobs with different priority levels; we represent such a priority by means of weights where the weight of a job typically represents the monetary penalty associated with the tardiness measure of a job. In practice, these could include tangible costs explicitly stipulated by contracts, intangible costs tied to loss of goodwill, or both. Also, as it is the case in practical applications, we assume that jobs can be interrupted so that higher priority jobs can be processed, and we will refer to this processing scheme as preemption.

We focus on the minimization of five tardiness related measures: total weighted tardiness (TWT), total weighted tardiness and overtime (TWTOT), total weighted earliness and tardiness (TWET), total weighted number of tardy jobs (TWNTJ) and total weighted completion times (TWC). We start our discussion in chapter 2 by presenting the relevant body of work that has been published in the area of optimization for tardiness related problems; we continue in chapter 3 by presenting the nomenclature and problem framework for all the tardiness related problems discussed. In chapter 4 we focus on a specific case of the TWT and TWTOT problems where these two tardiness measures are optimized for jobs constrained to equal processing times; the methods introduced in this chapter correspond to the two most conventional approaches used for the solution of these problems: heuristics and conventional binary models. We continue in Chapter 5 by taking the conventional approach in chapter 4 (the binary model) and compare its performance with an advanced aggregate model; again, this comparison is done using the TWT and TWTOT problems as test subjects. Finally we conclude our discussion by comparing the conventional and advanced modeling approaches this time under a more general umbrella of problems including all the tardiness related measures mentioned above for the most general versions of the problems; this is, with variable processing times.

CHAPTER 2

Literature Review

The problem of finding an optimal job schedule on a single-machine setting has been the center of attention of many research efforts in the last six decades; this has led to the creation of an extensive body of knowledge with a wide diversity of cases and extensions. For simplicity and correctness in the description of the problems here discussed, we use Graham's notation [1] as we survey the existing literature; under this notation every problem is expressed in a three-part description as in ($Machine$ | $Restrictions$| $Objective$) where the first part contains a number that indicates the machine environment *i.e.* number of machines - 1 in our case. The second part includes the restrictions and particular conditions under which the problem is defined, and the third part includes the objective function of the problem. Any restrictions on preemption, the processing time, release dates and due dates are commonly represented by $pmtn$, p, r and d respectively while the objective function may include w to indicate weights associated to any tardiness measure such as tardiness (l), completion time (C) or the number of tardy jobs (U) among others.

In the particular case of the single machine scheduling problem, a significant number of researchers consider total weighted tardiness (TWT) as a relevant and practical objective from which a diversity of tardiness related problems emerge. The

majority of published work focuses on non-preemptive scheduling [2, 3], where it is established that the general unrestricted TWT problem, $1 \mid\mid \sum w_i l_i$ is NP-hard in the strong sense [4], and based on the observation that every preemptive schedule can be transformed into a non-preemptive schedule with no larger objective value [5], the result due to [4] pertaining to complexity also applies to the general TWT problem with preemption [6, 7].

A few papers consider preemption for the single machine TWT problem in recent years. Most papers in this group study either tardiness with no consideration to weights [8–10] or weighted completion times [11, 12] under the total weighted completion problem (TWC); others, branch out into tardiness related problems involving more discrete measures of tardiness such as the total number of tardy jobs (TNTJ), while some branch out into composite measures of performance where the objective function involves a tardiness related measure along with other measures of performance such as overtime or earliness as in the TWT problem with overtime (TWTOT), or the total weighted earliness and tardiness problem (TWET). The topic of tardiness related measures as such, encompassing total tardiness (TT), TWT, the number of tardy jobs (NTJ) and the total weighted number of tardy jobs (TWNTJ) has only been studied in terms of their complexity and the polynomial equivalences between some of these problems [13]; leaving out of the research single methodological approaches that can provide timely exact solutions to the family of problems in this group.

For the case with equal length processing times and no weights ($1 \mid pmtn, r_i, p_i = p \mid \sum_i l_i$) [8] provide a polynomial time algorithm. In their subsequent work, the authors propose a pseudo-polynomial algorithm with unequal processing times for the

case where job release (r) and due dates (d) are agreeable, *i.e.*, $r_i < r_j$ implies that $d_i < d_j$. As for the TWT problem, the computational complexity with equal-length processing times is still open for both the non-preemptive [14] and the preemptive [15] cases.

Integrating the overtime option into the performance measures as in the TWTOT problem, has been studied by a number of researchers such as [16–21] under varying settings and objectives; however, only a few consider the weighted tardiness as part of the objective. The most closely related works are due to [17] and [20] who study the problem of minimizing a composite cost function of overtime and total weighted tardiness via heuristic approaches for a single resource setting. While both papers focus on non-preemptive schedules, we consider the more general case where preemption is allowed.

While preemptive scheduling literature is rather rare, non preemptive literature in the arena of composite performance functions is relatively abundant; In particular, with the advent of the "Just in time" paradigm (JIT) a few decades ago the problem of minimizing total weighted earliness and tardiness (TWET) became very popular, and received a good deal of research attention. The more general TWET problem where no machine idle time is allowed was researched by [22], and [23] where branch and bound algorithms and heuristics are proposed to address the problem. Other more specific cases of the TWET problem have been independently researched leading to a wide variety of subproblems branching out from the TWET. [24] proposes an exact procedure for the TWET problem under precedence constraints, and evaluates the effect of the arrangement of the due dates on the performance of the solution

method; [25] on the other hand, presents a metaheuristic for the TWET problem with common due dates where the due date is of restrictive character (*i.e.* when the due date is greater than the sum of all job processing times), and trade-off becomes a necessary element in the solution approach. Other efforts illustrating the diversity of research on composite performance functions have explored the TWET problem focusing on the case where the job weights are assigned in proportion its processing time [26], or where tardiness is restricted under weighted earliness [27].

Research using composite functions has also branched out to exploring mixed performance criteria involving weighted and non-weighted tardiness measures. [28] proposed a heuristic and compared it to an exact method using branch and bound to minimize the total weighted earliness subjected to the minimal number of tardy jobs. Weighted tardiness measures are common in the literature, and problems like the total weighted number of tardy jobs (TWNTJ) has also been studied, although not extensively and in non-preemptive settings. Early work on heuristics for the NTJ and TWNTJ introduced approximation algorithms that can be computed by hand [29]; more recent work [30] on multi-agent scheduling to minimize the total weighted number of tardy jobs introduced a multi-agent or customer oriented perspective into the arena of tardiness problems; here different agents (*i.e* customers) are used to represent independent interests in the optimization, and presents close resemblance to the paradigm that we present in the aggregation models. Although this work introduces the customers as important optimization elements it does not associate and group them via weights as we do.

In our research context the concept of trade-off has a paramount importance, and is constantly present in problems dealing with overtime capacity and tardiness costs

such as in TWTOT, this concept has also been studied as it relates to the resource allocation and the number of tardy jobs; [31] employs analytic methods to construct a tradeoff curve between capacity allocation and the number of tardy jobs while [32] on the other hand studies a similar situation relating to the tradeoff between the amount of periodic maintenance and the number of tardy jobs under a scheduling setting.

Finally, to conclude the survey of problems and methodologies found in the literature, we focus on the TWC problem. Although the TWC problem is not strictly a tardiness related problem, its structure is in deed closely related to the TWT problem; The work on exact methods for the TWC problem has received significant coverage in the research literature, but its extent has been limited to studying it in isolation from the TWT problem; [33] presented a branch and bound algorithm for the TWC problem under release constraints, and [34] built upon the existing work to produce a branch and bound algorithm with improved performance.

While the literature in the area of tardiness related problems is vast and very diverse, only a select minority of such work has a direct connection with the topics and problems discussed in this document; here we have made an attempt to provide a comprehensive survey of the applicable work in the area. From a methodological perspective the attention received by tardiness related problems has been focused on the development of heuristics and exact methods using branch and bound and dynamic programming; however, research focusing on modeling approaches and single exact methods that can be used for multiple tardiness related problems is virtually nonexistent. From an experimental perspective research on exact methods has been limited to instances of relatively small size (up to 50 jobs). As follows we will attempt to fill this gap to some extent and present a piece of research that introduces efficient

modeling approaches for tardiness related problems as well as a practical discussion on the implementation of such models on instances of more significant size (i.e. 160 jobs). In what follows, we will proceed to present the mathematical context, and formal versions of the tardiness related problems that we will discuss though the remaining of this document.

CHAPTER 3

Problem Setting and Notation

We consider the single machine preemptive scheduling on a variety of tardiness related problems. In general, our objective is to minimize a tardiness related measure F such as the total weighted tardiness (TWT), total weighted earliness and tardiness (TWET), total weighted completion (TWC), or the total weighted number of tardy jobs (TWNTJ); in the particular case of the TWT problem we also consider the objective of minimizing the TWT and overtime (TWTOT) as a special case where overtime is an available resource. Regardless of the problem at hand, the optimization objective is always subject to given release r and due dates d for n jobs with different amounts of work required (p_w) or with different processing times designated by the letters p or P depending on the model used. It is assumed that the time horizon (*i.e.* the schedule) is constituted by a set of discrete periods of equal length, where the time index is represented by t, $t \in \{1, 2, \ldots, T\}$. We represent this general optimization problem as: $1 \mid pmtn, r_i, d_i, p_i \mid F$.

We let J represent the set of customers and I denote the set of jobs, where each customer j and each job i is assigned a weight w_i, a release time r_i, a due-date d_i and a processing time p_i. If job i is not completed by its due-date, it is considered tardy and its tardiness (positive lateness) is captured by l_i. In other cases where

earliness is considered is denoted by ϵ. Tardy jobs incur costs proportional to their tardiness, namely, $w_i l_i$. In this context, the weights represent the unit tardiness costs for the jobs negotiated beforehand with the associated customer. In the context of preemptive scheduling, the single processor (machine) cannot process multiple jobs concurrently, however, execution of jobs can be preempted. That is, processing of a job can be interrupted or temporarily suspended at any moment in order to allow for another job's processing. In general, unlimited preemption is allowed. It is assumed that there is no precedence relationship among jobs and the process starts for jobs are only constrained by their own release times.

We map the maximum amount of work that can be performed on a job during regular time at each period to α_x, which measures the fraction of a job's required work. For example if $p_w = 1$, $\alpha_x = 0.4$ indicates that if we assign the full capacity to a particular job at a given period, 40% of that job can be completed. This implies that the processing time, p, for the job is 2.5 periods. As such, the processing times are allowed to take fractional values. Here, α_x can take values greater than p_w, representing the cases where the processor can complete more than a single job's load in a period. For the TWT problem in particular we allow additional limited and costly overtime capacity that can be employed at each period. Likewise, we characterize the overtime limit (α_v) as the maximum amount of a single job's work that can be completed on a single period. It should be noted for $\alpha_v = 0$, the problem is reduced to a special case where there is no overtime option (TWT). We let x_{it} or x_{jit} be a continuous positive variable that represents the regular time capacity assigned to job i from customer j on period t; similarly, we let v_{it} or v_{jit} be the fraction of job

i in customer j assigned to overtime capacity in period t. Clearly, both x_{jit} and v_{jit} take values in $[0, 1]$.

Basically, the overtime usage decision is driven by the trade-off between the tardiness and overtime costs. We assume that regular time cost is fixed over the horizon and does not depend on the workload assigned for regular time in a given period. In this context the regular time cost can be ignored. On the other hand, we let $\widehat{C}_v$ denote the cost of using all the available overtime in a period. Thus, the overtime cost in period t for given usage amount, v_{it}, can be computed as $(\widehat{C}_v/\alpha_v) \times v_{it}$. We let $c_v = \widehat{C}_v/\alpha_v$, which represents the overtime cost for each fraction of a job.

3.1 Problem Complexity

Problem complexity is a major area of research in Operations Research and the Computational Sciences; the size an importance of this area of research is a direct result from the vast diversity of problems existing and the broad impact that each one of these problems have; take for an instance the TWT problem; although it is generally considered a single problem, technically speaking, is a family of problems, and the complexity of the problems in this family can go from polynomial to NP-Hard just by changing a single problem parameter; this is, on one hand the non preemptive TWT problem for single machine with all job processing times equal to one period ($1 \mid r_i, d_i, p_i = 1 \mid \sum w_i l_i$) is of polynomial complexity [15], but on the other hand just by allowing the jobs to take different processing times as in $1 \mid \mid \sum w_i l_i$ the problem becomes NP-Hard [4, 35] . Similarly each one of the tardiness related problems discussed in this document represents a family of problems with diverse levels of complexity.

To keep our discussion within a manageable scope we focus only on the complexity of tardiness related problems concerning with single machine operations, preemptive allocation, weighted tardiness measures (F), and unrestricted release and due dates (*i.e.* $1 \mid pmtn, r_i, d_i, p_i \mid F$). Labetoulle [36] showed that the TWC problem ($1 \mid pmtn, r_i, d_i, p_i \mid \sum w_i C_i$)is NP-Hard, and others [4, 35] also showed that the non preemptive version of the TWT problem ($1 \mid r_i, d_i, p_i \mid \sum w_i l_i$) is also NP-Hard; Although in the literature the TWT problem with equal processing times ($1 \mid pmtn, r_i, d_i, p_i = p \mid \sum w_i l_i$) is currently of open complexity status [15], it can be seen from the existing result by [36] that the $1 \mid pmtn, r_i, d_i, p_i \mid \sum w_i l_i$ is fundamentally NP-Hard; this is, by decomposing the weighted tardiness measure $\sum w_i l_i$ as

$$\sum w_i l_i = \sum w_i (f_i - d_i) \tag{3.1}$$

where $f_i = C_i + r_i - 1$. Here we see that the resulting tardiness expression in terms of C_i is composed of a variable and three constant terms as in

$$\sum w_i l_i = \sum w_i C_i + \sum w_i r_i - 1 - \sum w_i d_i \tag{3.2}$$

from here it becomes apparent that the solution to the above TWT problem is equivalent to the solution of its related TWC problem plus three constant terms ($\sum w_i r_i, -1, -\sum w_i d_i$). Under this reasoning we can see that the previous work on the complexity of the TWC in [36] constitutes also a tool to classify our TWT problems also as NP-Hard. Similarly, the same reasoning can be used to show that the related TWET problem is NP-Hard; this is, if we allow the tardiness expression in $f_i - d_i$ to take negative values and evaluate the weighted tardiness and earliness as

$|C_i + r_i - 1 - d_i| = l\epsilon_i$ we can see that it results in an equivalent tardiness measure as in the TWT problem ($\sum w_i l\epsilon_i$)with an equivalent level of complexity to the TWC and TWT problems; also, on a side note, in the case where preemption is not allowed this same problem (TWET) was shown to be NP-Complete by [38].

Finally, the TWNTJ problem as in $1 \mid pmtn, r_i, d_i, p_i \mid \sum w_i U_i$ and the TWTOT problem as in $1 \mid pmtn, r_i, d_i, p_i \mid \sum w_i l_i + c_v \sum \sum v_{it}$ are problems currently with open complexity status; although in the case of the TWNTJ [37] showed that its non preemptive version is NP-Hard in the ordinary sense; nonetheless, through a similar exercise as for the TWET it can be seen that the TWNTJ problem has also NP-Hard character; we can see this by expressing the binary tardiness measure U as

$$U_i = \begin{cases} 1: & \text{if } (C_i + r_i - 1 - d_i) > 0 \\ 0: & otherwise \end{cases} \tag{3.3}$$

and noting that its non zero values correspond to a solution of the TWT problem. Now for the case of the TWTOT, its NP-Hard complexity character can be seen from the interactions between the weights w_i and the cost of overtime c_v; here we see that when the cost of overtime c_v is significantly larger than the cost of tardiness w_i the optimization neglects the use of overtime and the problem turns into solving its related TWT problem; on the other side when the cost of tardiness is significantly larger than the cost of overtime ($w_i >> c_v$) the optimization will resort to use overtime by default, and the capacity allocated will be enhanced by the extra overtime capacity while the problem at heart is still the TWT. In all other cases where the costs of tardiness and overtime are comparable the TWTOT requires the evaluation of the cost trade-off which by its nature is more complex than any of the two previous extreme cases for w_i and c_v.

CHAPTER 4

Heuristics and Conventional Models

4.1 Overview

In this chapter we study two proposed heuristics and the conventional modeling approach for the solution of the Total Weighted Tardiness problem (TWT) with and without overtime capacity allocation. In particular we focus on the scheduling of a finite set of jobs on a single resource that operates under preemption, and have associated release and due dates. Limited overtime capacity can be utilized to reduce tardiness; however, since overtime is costly, justification of the overtime use depends on the trade-off between the tardiness and the overtime costs. The overall objective of the heuristics and the conventional model is to minimize the total cost of tardiness and overtime.

We first propose a heuristic solution for the the TWT problem that allocates workload to available capacity based on a dynamic priority rule. Basically, the priority of a job at a given time depends on its weight (tardiness cost), due date, and the remaining workload. Later we generalize our analysis to the setting that allows for availability of limited overtime capacity. Under a holistic approach a three-stage heuristic is developed for minimizing the total cost of tardiness and overtime utiliza-

tion. In the first stage, overtime is treated as part of the regular capacity. In other words, it is assumed that overtime is not costly. The priority rule developed for the base setting is employed to generate an initial non-delay schedule. In the second stage, the overtime usage is reduced by shifting workload and generating a full-delay schedule without altering the tardiness of jobs produced in the first stage. In the final stage, the heuristic attempts to improve the total costs by means of extending the tardiness of jobs in return for reduction in overtime utilization resulting with net gains. At the end, the three-stage procedure builds a full-delay schedule that allocate workloads among regular and overtime capacity slots based on the critical trade-off between the tardiness and overtime costs.

Both heuristics are tested using numerical analyses, and the conventional model (Exact MIP model) is employed as the benchmark to evaluate the performance of the heuristics. Using computational tests, we compare the performance of our heuristics with the upper bounds generated by the conventional model formulation. The results show that the proposed methods are efficient in obtaining good quality solutions in significantly short times and they can be useful as effective solution tools especially for large size problems.

4.2 The Conventional Mathematical Model

The exact model used in this research aims to find an optimal schedule that minimizes the total cost of tardiness and overtime for a given set of jobs by determining regular and overtime work allocations, finish dates, and tardiness. The nomenclature for the decision variables is given in Table 4.1.

Table 4.1: Decision Variables of the Exact Model

x_{it}	Fraction of a job i assigned during regular time at period t
v_{it}	Fraction of a job i assigned during overtime at period t
y_{it}	Binary variable: 1 if job i finishes on period t, 0 otherwise
f_i	Finish date for job i
l_i	Number of late periods on job i

The mathematical model involves binary, continuous, and discrete decision variables. The binary variables y_{it} track the completion time of jobs. While the continuous variables determine the amount of allocations of regular capacity x_{it} and overtime work v_{it} that are necessary to complete the jobs, the discrete variables are needed to evaluate the finish date f_i as well as the number of late periods l_i. The tardiness cost for job i is captured by $w_i l_i = w_i \times max\{0, f_i - d_i\}$. On the other hand, the overtime capacity cost for job i in period t is given by $c_v v_{it}$. We can write down the mathematical model as follows:

$$\text{minimize} \quad \Omega = \sum_{i=1}^{n} w_i l_i + c_v \sum_{t=1}^{T} \sum_{i=1}^{n} v_{it} \tag{4.1}$$

subject to:

$$\sum_{t=r_i}^{T} y_{it} = 1, \qquad \forall i \in I \tag{4.2}$$

$$f_i = \sum_{t=1}^{T} t y_{it}, \qquad \forall i \in I \tag{4.3}$$

$$l_i \geq f_i - d_i, \qquad \forall i \in I \tag{4.4}$$

$$\sum_{i=1}^{n} x_{it} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{4.5}$$

$$\sum_{i=1}^{n} v_{it} \leq \alpha_v, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{4.6}$$

$$\sum_{t=r_i}^{T} (x_{it} + v_{it}) = 1, \qquad \forall i \in I, \tag{4.7}$$

$$\sum_{t=r_i}^{\gamma} (x_{it} + v_{it}) \geq y_{i\gamma}, \qquad \forall i \in I, \forall \gamma \in \{r_i, \ldots, T\}, \tag{4.8}$$

$$x_{it},\ v_{it} \in [0, 1]\,; \quad f_i,\ l_i \in \mathbb{Z}^+; \quad y_{it} \in \{0, 1\} \tag{4.9}$$

The objective function given in (4.1) minimizes the total cost of tardiness and overtime capacity. As mentioned above, the weight for job i, w_i, maps the unit cost of tardiness for that job in this representations. The first set of constraints (4.2) ensure that each job is completed by the end of the planning horizon. The second set of constraints (4.3) capture the completion times. Given completion times, constraint in (4.4) sets the tardiness for the jobs. Constraints (4.5) and (4.6) enforce the regular and overtime capacity limits. Constraint (4.7) allocates all work required to complete a job across time periods following its release date and (4.8) ensures that jobs are not completed before all required work is done. Solution to the above model provides

an optimal preemptive schedule for a given set of jobs with different release and due dates.

4.3 Total Weighted Tardiness (TWT) Heuristic

In this section, we first introduce an efficient heuristic method for the base case, where overtime capacity is not explicitly considered in the problem. That is, we assume either no overtime capacity is available or it does not incur any additional cost. In this case, the problem reduces to the conventional single machine scheduling problem, *i.e.*, $1|pmtn, r_i, p_i = p|\sum_i w_i l_i$. The solution approach for the general case will later be built on this heuristic in the following section. The proposed heuristic dynamically prioritizes the jobs based on their weights, remaining processing times (RPT), and remaining available times (RAT). It allocates available capacity (α_x) to the highest ranked job from the subset of jobs that have been released and have not been completed (I_R). Ranking is updated at the beginning of each period or at the time a job is completed based on the priority rule calculated by $w_i/max\{RPT_{it}, RAT_{it}\}$ for each released job i. Higher values signify higher priorities for the jobs. In what follows, we present the formal definitions for the components of this priority rule and rationalize its effectiveness on minimizing the total weighted tardiness for the preemptive scheduling problem.

Definition 1 ***Remaining Available Time (RAT)*** *is the time to due date for a job at any given period in a given partial schedule. Specifically,* RAT_{it} *represents the remaining time for job* i *as of time* t *until its due date and as such,* $RAT_{it} = max\{0, d_i - t + 1\}$.

Definition 2 ***Remaining Processing Time (RPT)*** *is the amount of remaining time required to complete the processing of a job in a partial schedule. Specifically, for job i at the beginning of period t, $RPT_{it} = (1 - \sum_{z=r_i}^{t-1} x_{iz})/\alpha_x$.*

We note that at period t, both RAT_{it} and RPT_{it} are defined only for any job i such that $r_i \leq t$ (*i.e.*, job is released) and $f_i \geq t$ (*i.e.*, job is not completed yet).

Definition 3 ***Remaining Slack (RS)*** *is the slack time for a given job at a given period in a given partial schedule and calculated by the difference between the RAT and the RPT. Specifically, for job i at time t, $RS_{it} = max\{0, RAT_{it} - RPT_{it}\}$.*

Based on above definitions, we can rewrite the priority of job i at time t, Ψ_{it}, as follows:

$$\Psi_{it} = \frac{w_i}{RPT_{it} + RS_{it}} \tag{4.10}$$

It is clear in (4.10) that slack time (RS) influences a job's priority only when it is strictly positive. When $RS_{it} = 0$ in a partial schedule, it is realized that at time t job i cannot be completed before its due date. At his point, the priority is a function of the job's weight and RPT. To explain the priority rule for this instant, consider the general case where all jobs will be tardy, *i.e.*, $f_j \geq d_j$ for all j. In this case, the objective function for these jobs can be rewritten as

$$\sum_j w_j l_j = \sum_j w_j (f_j - d_j) = \sum_j w_j f_j - \sum_j w_j d_j \tag{4.11}$$

We note that since due date values are exogenous, the optimization reduces to the minimization of $\sum_j w_j f_j$. This observation leads to the following conclusion:

Corollary 1 *When the due dates are sufficiently tight so that no job can be completed before its due date, the Total Weighted Tardiness (TWT) problem reduces to the Total Weighted Completion (TWC) problem.*

This relationship between the TWT problem and the TWC problem is significant in that it is shown by [39] that priority rule w_i/RPT_{it} results in an optimal local ranking among released jobs in a partial preemptive schedule. Consistently, our method is reduced to this rule when $RS_{it} = 0$ for all $i \in I_R(t)$. In this context, local optimality at time t is signified by a schedule among available jobs which makes the smallest possible contribution to the objective function provided that all other jobs released after time t are ignored.

On the other hand, when $RS_{it} > 0$, the slackness must be exclusively included in the priority rule so as to incorporate the due date factor. To rationalize the inclusion of positive RS, we first make the following observation:

Proposition 1 *Consider a partial schedule at time t, where for all available jobs the remaining slack is strictly positive. That is, $RS_{jt} > 0$ for all $j \in I_R(t)$. The ranking obtained based on the priority rule given in (4.10) does not violate local optimality.*

Proof: Suppose $\Omega(I_R(t))$ represents the locally optimal total weighted tardiness among all jobs in $I_R(t)$. Moreover, let k represent the index for the job with the highest priority in $I_R(t)$ calculated based on equation (4.10) at time t. To make the proof, it is sufficient to observe that if we delay the preemption of job k by a sufficiently short time interval, say δ time units, where $I_R(t) \equiv I_R(t+\delta)$, $\Omega(I_R(t))$ can still be obtained. This is due to the fact that unlimited numbers of preemption are allowed. ■

The above result is demonstrated in figure 4.1. Suppose job k is released at r_k and has higher priority at that time. As such it preempts job i based on the priority rule (Schedule A). At this point, jobs i and k have strictly positive slacks, namely, $d_i - r_k - 2 = 1$ and $d_k - r_k = 6$ respectively. Alternatively, if we delay the preemption point by one period (Schedule B), same total weighted tardiness can still be attained since, at this point, both jobs still have strictly positive slacks ($d_i - r_k - 2 = 1$ and $d_k - r_k - 1 = 5$ respectively).

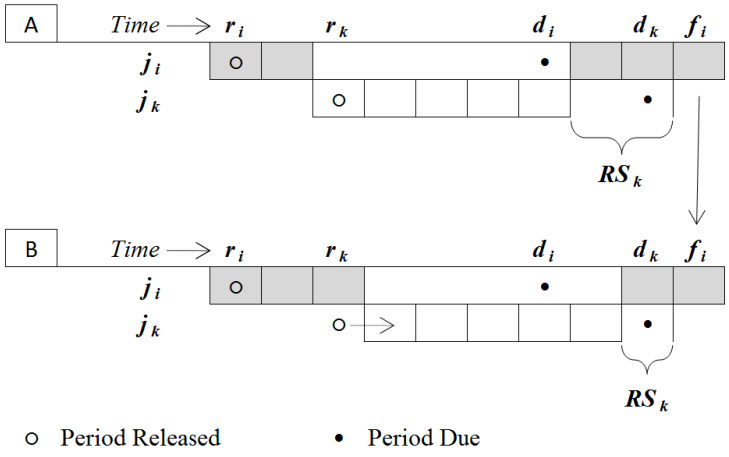

Figure 4.1: Schedule A - Preemption at r_k, Schedule B - Preemption at $r_k + 1$

As indicated by Corollary 1 and Proposition 1, the critical part of the priority ranking by heuristic is predicated upon the comparisons between the jobs with no slack and those with positive slack. On one hand, a job may have no slack but more remaining work. On the other hand, a job with less remaining work may have a large amount of remaining slack time until its due date. Clearly, the proposed priority rule attempts to efficiently capture the interplay between such jobs. The joint impact of the RPT and RAT on the priority level of a job, which has no capacity allocation until period z, is illustrated in Figure 4.2 across time periods.

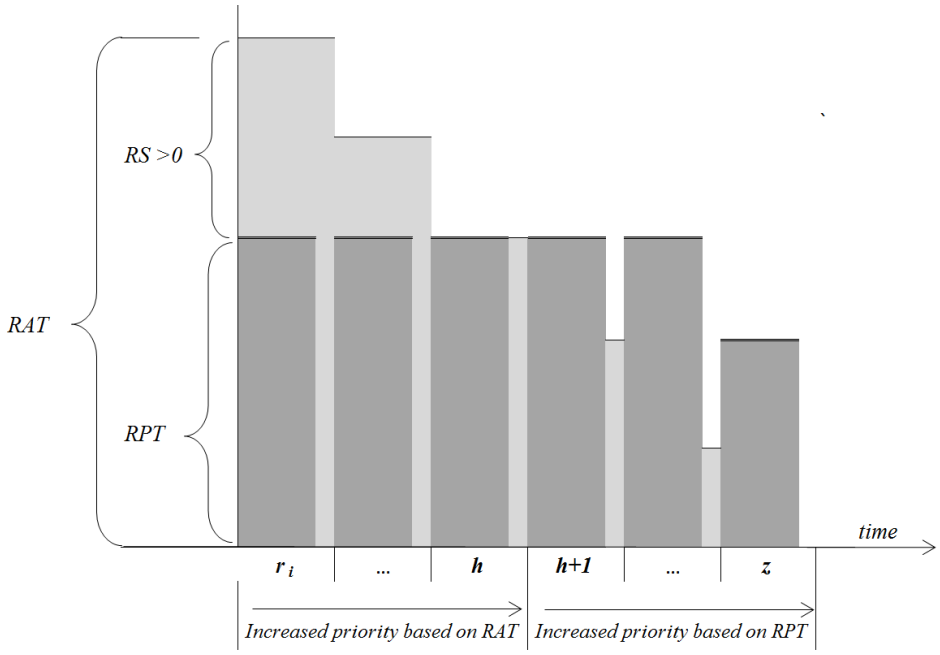

Figure 4.2: Illustration of the priority rule $w_i/max\{RPT_{it}, RAT_{it}\}$

4.3.1 Computational Experiments

We design and perform numerical analysis so as to evaluate the computational performance of the TWT Heuristic. We aim to determine the computational performance of the heuristic with respect to the exact method as the number of jobs changes. Therefore, in our experiment, we include a varying problem sizes. Specifically, we consider seven scenarios based on problem sizes of 7, 10, 15, 20, 30, 40, and 80 jobs.

We use a set of expressions to generate all the parameters necessary to create the problem instances. To be consistent with the existing literature, we borrow the expressions from earlier work due to [20] and modify them to our context. Main random variables that are used to generate the parameter space follow uniform distributions in order to enable diversity across all instances. Table 4.2 lists the specific parameter generating expressions used in our analysis. For each scenario, 20 instances were

generated. As a result, a total of 140 instances are obtained to test the performance of the heuristic and the exact model. All instances were solved using a PC with an I7-3770 Processor with 3.40 GHz and 16 GB of RAM.

Table 4.2: Parameter Generating Expressions

Parameters	Expressions
Time horizon - T	$r_{last} + n/\alpha_R$
Release date - r_i	UNIF$(1, n)$
Due date - d_i	$r_i + 1/\alpha_R$+UNIF$(0, a_2)$
Tardiness penalty (weight) - w_i	UNIF$(1, a_1)$
Regular time capacity - α_R	UNIF$(0.1, 0.9)$
	$a_1 = 2n/3$, $a_2 = 2$

For the exact model given in Section 4.2, CPLEX 12.0 is used as the solver. However, due to the computational complexity, not all the instances lead to exact solutions in reasonable amounts of time. Therefore, we set a fixed time limit for the execution of the exact model. Specifically, the computational runs are set to initiate a time-out sequence at 7200 seconds (2 hours). Once the time-out sequence is initiated, we observe that it takes about additional 3 minutes in average to finalize the current iteration. In cases where exact solutions are obtained, we contrast these optimal solutions to the heuristic solutions, and in cases where the exact solution is not reached, we compare the best feasible solution converged by the solver with the solution provided by the heuristic.

4.3.2 Computational Results

We investigate the results of all 140 instances (20 for each problem size) on each solution approach totaling 280 runs. In our numerical analysis, we divide the solution sets into two groups. In the first group, we evaluate the cases where the optimal solutions are obtained by the exact model within the given time frame. The instances in this case make 31% of all the instances (*i.e.*, 43 of the 140 instances) all corresponding to the problem sizes with 7, 10 and 15 jobs. In 34 of these 43 instances, the TWT heuristic also produced the optimal solutions. In general, the heuristic achieved near optimal solutions with an average gap of 1.07%. The computational performances are summarized for this case in Table 4.3.

Table 4.3: Group 1 - Optimal Instances

	Exact Model				Heuristic					
	Comp. Time (sec)				Comp. Time (sec)			Percent Error (%)		
Jobs	OPT	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
7	20	0.209	22.32	172.1	0.004	0.005	0.008	0.00	0.29	3.33
10	14	2.46	294.4	3208	0.001	0.003	0.030	0.00	1.55	7.51
15	9	2.887	1792	4989	0.005	0.006	0.008	0.00	1.36	7.41

Unfortunately, optimal solutions cannot be reached within the given time frame for the rest of the instances; these are assigned to the second group of solution sets. While the exact model reported optimal solutions for most of the instances of the smaller problem sizes (7 and 10 jobs), all of the instances for problem sizes 20 and above did not converge to optimality before the 2-hour time-out threshold. Due to lack of the optimal solutions and good quality lower bounds, we compare the results of our heuristic with the timed-out (TO) solutions obtained by the exact solution method for this group. Overall, in this group, the proposed heuristic resulted with

smaller weighted total tardiness values compared to those obtained by solver in 77 of the 97 instances (79%). Specifically, the heuristic resulted with better solutions in all instances for problem sizes 40 and 80 jobs. The summary of the results are provided in Table 4.4.

Table 4.4: Group 2 - Timed-Out Instances

	Exact Model				Heuristic					
	Comp. Time (sec)				Comp. Time (sec)			Percent Difference (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max	Worst	Avg	Best
10	6	7207	7229	7326	0.001	0.002	0.004	0.05	0.01	0.0
15	11	7206	7766	9034	0.006	0.020	0.112	0.80	-0.13	-1.72
20	20	7203	7719	9062	0.002	0.006	0.011	7.00	-0.65	-5.63
30	20	7203	7876	8544	0.004	0.009	0.017	3.10	-2.54	-7.20
40	20	7207	7864	8468	0.006	0.015	0.040	-1.88	-4.20	-7.71
80	20	7202	7711	9039	0.010	0.020	0.050	-4.49	-16.1	-40.2

We observed that as the problem size increases, the convergence performance degrades significantly for the exact model. The average optimality gap stays above 44% for more than 20 jobs with a worst case performance of 89.1% gap. Consequently, as the problem size increases the quality of the heuristic solution shows an increasing improvement over the timed-out exact model. On average, the TWT heuristic generates solutions with 0.13%, 0.65%, 2.54%, 4.24%, and 16.16% reduction in total weighted tardiness in comparison to the best feasible solutions obtained by the exact model after two hours of run time for 15, 20, 30, 40, and 80 jobs respectively. A summary of the relative improvement of the heuristic with respect to the timed-out exact model is depicted in Figure 4.3.

In terms of computational times, the gap is significant between the two cases. For the 40 instances with 7, 10 and 15 job problems, the average computational time for

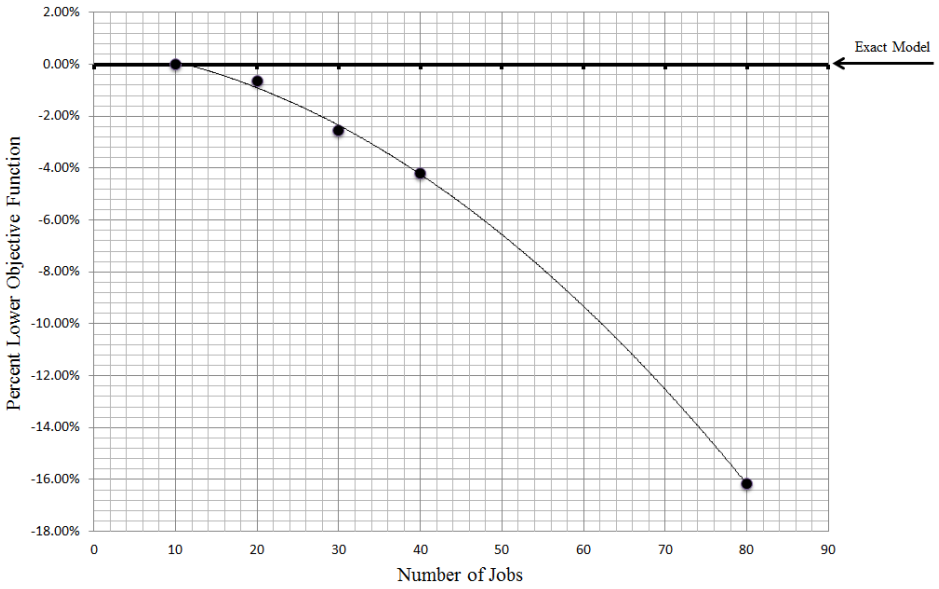

Figure 4.3: TWT heuristic average performance vs. exact model

obtaining the optimal solution with the exact solution method is 702 seconds (about 5 minutes) on average with a maximum value of 83 minutes. On the other hand, it takes only a small fraction of a second for the heuristic to produce a solution. In general, the solutions are reached within time frames of milliseconds in all instances by the TWT heuristic as summarized in Tables 4.3 and 4.4.

4.4 Total Weighted Tardiness and Overtime (TWTOT) Heuristic

The presence of overtime option offers opportunities to reduce the total tardiness in a schedule. However, typically, this option is costly and limited. As such, overtime capacity and associated cost must be explicitly incorporated into the scheduling process, which aims at optimally balancing the costs due to tardiness and overtime usage. Here, we present a holistic procedure that achieves this. The proposed procedure is

built on the concepts and observations related to the TWT algorithm introduced in the previous sections. For the TWTOT problem at hand, we adopt the *compact and relax* principle introduced by [20] to solve the non-preemptive version. In this approach, ignoring the overtime cost, a non-delay schedule is built using all available overtime capacity as needed in the initial (compact) phase. In the second (relax) phase, the initial schedule is "relaxed" by shifting the schedule to the right and thus, reducing the overtime usage subject to cost improvements.

Although the principle employed in our approach is similar to that of [20], the specifics of the proposed procedure are quite different. The difference is mainly due to the presence of the preemption option for all jobs, which does not necessarily lead to a strict sequence of jobs. First, we build our initial schedule based on the TWT heuristic developed particularly for the preemptive scheduling problem. Second, in contrast to [20], our "relax" phase is not built based on a fixed sequence of jobs. Clearly, in a preemptive schedule, a fraction of a job can be shifted beyond other scheduled jobs. This requires a procedure that must efficiently evaluate a higher number of different "relaxing" options.

The proposed TWTOT heuristic is implemented in three stages. The first stage corresponds to the "compact" phase where an initial schedule is generated via the TWT heuristic utilizing all available overtime capacity as needed. The second and third stages compose the "relax" phase. The second stage generates the schedule that minimizes the total overtime cost under the tardiness constraints imposed by the initial schedule. Finally, the third stage incorporates the tardiness and overtime cost trade-offs and evaluate schedule improvements that reduce the total cost of tardiness and overtime usage. In what follows the details of these stages are discussed.

Stage 1: Initial Schedule

As mentioned above, an initial schedule is constructed in this stage by employing the TWT heuristic introduced in the previous section. The TWT heuristic in this stage does not assume any cost for the overtime capacity. Therefore, the regular and overtime capacities are indistinguishable and they are simply stacked together in a given period. Consequently, the per period capacity is updated by $\widehat{\alpha}_x = \alpha_x + \alpha_v$ before the TWT heuristic is implemented as described in the previous section. In the rest of the chapter, we let Ω_{TWT} denote the value of the objective function (4.1) obtained at the end of this stage.

Stage 2: Full-Delay Schedule

In this stage, we attempt to reduce overtime usage as much as possible without impacting the tardiness of jobs determined by the initial schedule. In other words, the goal is to come up with an alternative schedule that achieves the same total weighted tardiness with minimum overtime capacity usage. To achieve that, we modify the non-delay schedule generated by the TWT heuristic into a full-delay schedule, where the jobs are delayed as much as possible without violating their first stage tardiness outcomes. A full-delay scheduling approach is introduced by [40] to minimize the total earliness with hard deadlines and no preemption in a different context. We employ a modified approach tailored to our context where preemption is allowed. We replace the due dates of the jobs with the maximum of their original due dates and finished dates obtained by the TWT heuristic. That is, for job i, $\widehat{d_i} = max\{d_i, f_i\}$. The schedule then is shifted towards the available downstream regular time slots so as to reduce the overtime usage while disallowing any tardiness with respect to the

updated due dates. To explain the details of this process, we first need to introduce the following definition:

Definition 4 *Resulting Regular Time Slack (RTS) is the amount of unused regular time capacity that can be assigned to a job without altering its tardiness or any other job's tardiness in a given schedule. For any job i,*

$$RTS(i) = (\widehat{d}_i - f_i + 1)\alpha_x - \sum_{t=f_i}^{\widehat{d}_i} \sum_{j \in \boldsymbol{I}} x_{jt} \tag{4.12}$$

To help understand what actually RTS captures, consider the schedule given in Figure 4.4. We note that in this schedule, for ease of presentation, each slot represents capacity that can process one ninth of a job. As such, $\alpha_x = 1/3$ and $\alpha_v = 1/9$. Suppose that job 2 is originally due by the end of period 12 and both jobs 3 and 4 are released in period 7 with due dates of 10. Here, it is clear that $f_2 = 6$, $f_3 = 9$, and $f_4 = 11$. Then, applying (4.12), the updated due dates become $\widehat{d}_2 = 12$, $\widehat{d}_3 = 10$, and $\widehat{d}_4 = 11$. To calculate $RTS(2)$, first visually observe that there are 5 available regular time slots between the period in which job 2 is completed and the period at which the job is due. Other slots are allotted to jobs 3 and 4. As such, $RTS(2) = (12 - 6 + 1)/3 - 16/9 = 5/9$ implying that there is available regular time capacity enough to process five ninths of job 2 before its deadline. For job 4, $RTS(4) = (11-11+1)/3-2/9 = 1/9$ implying that the slack for this job is equivalent to one ninth of its required workload. For job 3, on the other hand, $RTS(3)$ is clearly 0 because there is no unused regular capacity between periods 9 (f_3) and 10 ($\widehat{d}_3$).

Capturing RTS is significant as it gives us indications for potential overtime reduction. For example, in the example given in Figure 4.4, we realize that overtime use for jobs 1, 2, and 4 can be reduced by shifting workloads from overtime capacity

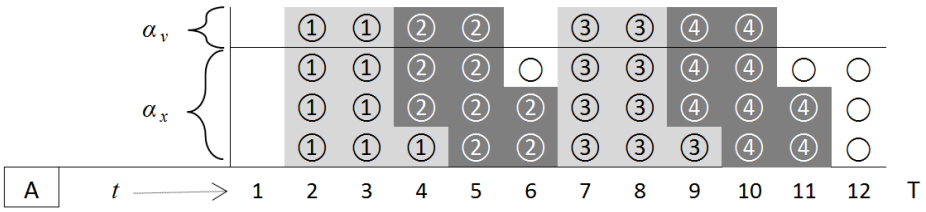

Figure 4.4: Regular Time Slack (RTS)

to available regular capacity without increasing tardiness of any job. Specifically, $\sum_i RTS(i)$ gives us the upper bound on overtime reduction that can be achieved without delaying jobs beyond their updated due dates. This value is used as a criterion in our full-delay (FD) algorithm. We let I_t denote the set of jobs for workload scheduled in period t and T_L denote the last period with any scheduled workload. The steps of the FD algorithm are presented in Table 4.5.

Table 4.5: The Full-Delay Algorithm

Step 0	$t \leftarrow T_L$.
Step 1	If $t = 0$ stop. Else, compute $RTS(i)$ for all $i \in \mathbf{I}_t$.
Step 2	Identify the job with the highest RTS value in I_t: $j \leftarrow \arg\max_{i \in \mathbf{I}_t} RTS(i)$.
Step 3	If $RTS(j) = 0$ or $\mathbf{I}_t = \{\emptyset\}$ then $t \leftarrow t - 1$ and go to Step 1. Else, starting with all current overtime allocations, wherever feasible, shift all upstream workload to all unused downstream regular capacity on or before $\widehat{d}_j$. Go to Step 1.

To illustrate the algoritm, consider again the non-delay schedule in Figure 4.4. The FD algorithm starts with period 11, where only job 4 is scheduled with $RTS(4) = 1/9$. Positive RTS indicates that workload related to this job can be shifted to the right. It should be noted that primarily all overtime workloads related to this job must be shifted to unused regular capacity. And the shift cannot extend beyond $\widehat{d}_4 = 11$. This results with the schedule given in Figure 4.5. We observe that in the new schedule while RTS becomes 0 for jobs 3 and 4, and $RTS(2) = 4/9$. Consequently, executing

$t = 6$, the algorithm generates the schedule in Figure 4.6. The algorithm results with an overtime reduction of four periods at the end.

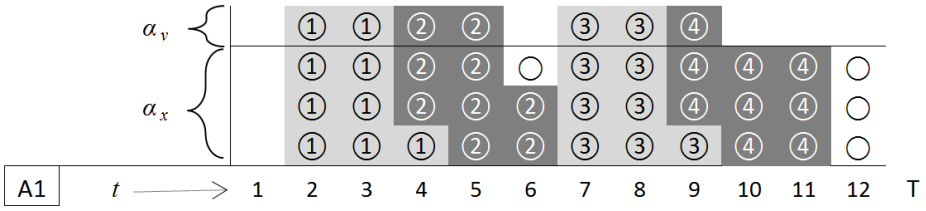

Figure 4.5: Schedule after first pass of the FD algorithm

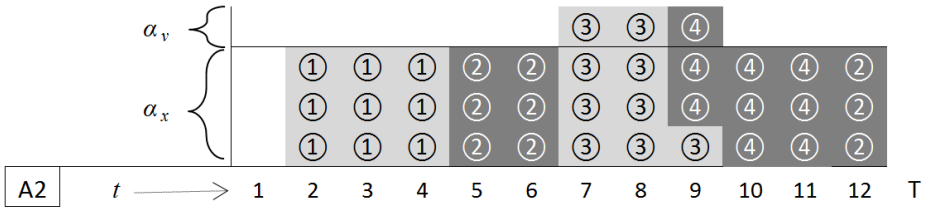

Figure 4.6: Final schedule generated by the FD algorithm

We let Ω_{TWT}^{FD} denote the value of objective function produced by applying the FD algorithm to the compact schedule generated by the TWT algorithm and make the following conclusion:

Proposition 2 *$\Omega_{TWT}^{FD} \leq \Omega_{TWT}$ always holds. Moreover, assuming that satisfying the updated due date is a hard constraint for each job, Ω_{TWT}^{FD} is optimal.*

Proof: The first part directly follows from two observations: First, the TWT heuristic does not distinguish overtime capacity from regular capacity and generates a non-delay schedule. Second, the FD algorithm maintains the same total weighted tardiness cost and attempts to reduce the overtime usage. For the second part of the proposition, note that the FD algorithm generates a full-delay schedule where RTS values for all jobs are zero. As such, any reduction in overtime usage is possible only if at least one job is completed after its updated due date. ■

Stage 3: Tardiness Relaxation

While the TWT heuristic of Stage 1 ignores the overtime costs and focuses on minimizing the total tardiness cost, the FD algorithm in Stage 2 concentrates only on reducing the total overtime cost under fixed tardiness costs. To complete the loop, Stage 3 aims to minimize the sum of both costs predicated upon the cost trade-offs between the tardiness and the overtime costs. The full-delay schedule obtained in the previous stage will be the initial schedule in this stage. As in the previous cases, we introduce key definitions that help us build the tardiness relaxation procedure in this stage.

Definition 5 *A Local Full-Delay (LFD) Block is defined as a schedule block in which there is no unused regular time capacity and for every job with allocated workload in this block RTS is zero.*

We note that a LFD block is only relevant for a full-delay schedule. To illustrate, consider the schedule in Figure 4.7. Suppose release periods are 2, 2, 3, and 6 for jobs 1-4 respectively. Moreover, the weights are 3, 4, 2, and 7, and updated due dates are 5, 4, 4, and 7 respectively. Observe that this is a full-delay schedule. Based on the above definition, jobs 1, 2, and 3 form a LFD block. Job 4 is not in this block because there is an unused regular capacity slot between this job and the others. As such, it forms a separate block. We let $\mathbf{E}$ represent the set of local full-delay blocks and $\mathbf{E}_k$ represent the set of jobs in block k. Hence, in the given example $\mathbf{E} = \{\{(1, 2, 3), 4\}$, $\mathbf{E}_1 = \{1, 2, 3\}$, and $\mathbf{E}_2 = \{4\}$. We also let $\widehat{t}_{k,0}$ and $\widehat{t}_{k,1}$ be the beginning and ending periods for block k.

Proposition 3 *All workload associated with any given job can be enclosed by exactly one LFD block in a full-delay schedule.*

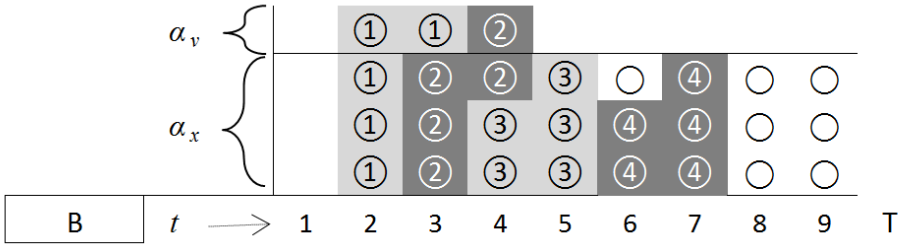

Figure 4.7: Local Full-Delay Blocks

The proof is straightforward from the fact that if a job is split into multiple blocks, the workload scheduled in an upstream block can be shifted to the right which is not possible in a full-delay schedule. The above observation reveals that LFD blocks are mutually exclusive in that a job exists in exactly one block. This helps us with our next definition:

Definition 6 *Tardiness-Overtime Payback (TOP) measures the maximum possible net gain in the total cost function by increasing the tardiness of a job which results with reduction in overtime usage. Specifically, $TOP_n(i)$ represents the upper bound on net gain by increasing the tardiness of job i by n periods in block k, where,*

$$TOP_n(i) = c_v \min\{(\alpha_x n - \sum_{t=\widehat{d}_i+1}^{\widehat{d}_i+n} \sum_{j \in \boldsymbol{E}_k} x_{jt}), \sum_{t=\widehat{t}_{k,0}}^{f_i} \sum_{j \in \boldsymbol{E}_k} v_{j,t}\} - w_i n \tag{4.13}$$

The TOP value is useful in assessing the trade-off between tardiness and the overtime usage. As reflected in (4.13), the upper bound on overtime reduction implied by a job in LFD block is limited by the minimum of the total overtime usage in the interval that spans from the beginning of the block to the completion time of the job and the regular time capacity that becomes available after the job's tardiness is increased. If the potential reduction is larger than the cost of the increase in tardiness, *i.e.*, if TOP is positive, then delaying this job should be considered. Recall the

schedule in Figure 4.7. Suppose that $c_v = 15$. Then, in the second block, $TOP_1(4) = -7$, which indicates that there is no potential gain in increasing the tardiness of job 4. On the other hand, in the first block, $TOP_1(3) = 3 - 2 = 1$. Since the TOP value is positive for this job, there is a potential to reduce the total cost by delaying it. Basically, the tardiness relaxation (TR) algorithm developed for this stage builds on this principle.

The TR algorithm employs a backward pass and begins with the latest LFD block. A backward pass is preferred because a shift in a downstream block has potential to open up more unused regular capacity for the upstream blocks. At each pass, the algorithm computes the TOP values for all jobs in the current block. If all the TOP values return negative values no tardiness relaxation is attempted. Otherwise, we let $\mathbf{E}_k^+$ represent the set of all jobs in block k with strictly positive TOP values. Using the greedy approach, the tardiness option with the highest positive payback is selected from this set each time to evaluate tardiness relaxation. Each time an increased tardiness is applied, the algorithm checks whether block formations change. This happens when two consecutive blocks mesh with each other and form a single block because the unused regular capacity gap is closed between the two following the tardiness relaxation. The process repeats until no improvement can be obtained in the objective function given in (4.1). The steps of the algorithm are presented in Table 4.6.

To illustrate the TR algorithm consider again the example in Figure 4.7. Recall that the updated due dates for jobs 1-4 are 5, 4, 4, and 7 respectively. The algorithm starts with the second and the last block. The only job in this block, namely job 4, has a TOP value of -7. Therefore, no tardiness relaxation is attempted for this block.

Table 4.6: The Tardiness Relaxation Algorithm

Step 0 Identify all LFD blocks. $M \leftarrow |\mathbf{E}|$; $k \leftarrow M$.

Step 1 Compute the TOP value of extending tardiness to period $\widehat{t}_{k,1}+1$ for each job i in $\mathbf{E}_k$.

Step 2 If $\mathbf{E}_k^+ = \{\emptyset\}$ or $\sum_{t=\widehat{t}_{k,0}}^{f_i} \sum_{j\in\mathbf{E}_k} v_{j,t} = 0$, then $k \leftarrow k-1$ and go to Step 4.
Else, $j \leftarrow \arg\max_{i\in\mathbf{E}_k^+} TOP_{\widehat{t}_{k,1}+1-\widehat{d}_i}(i)$; Go to Step 3.

Step 3 Apply the FD algorithm to the current schedule assuming $\widehat{d}_j = \widehat{t}_{k,1}+1$.
If the objective function improves, replace the current schedule with the new one; update $\widehat{t}_{k,0}$ and $\widehat{t}_{k,1}$ for LFD block k; $\widehat{d}_j \leftarrow \widehat{t}_{k,1}+1$.
Go to Step 1.
Otherwise, keep the current schedule; $\mathbf{E}_k^+ \leftarrow \mathbf{E}_k^+ - \{j\}$; Go to Step 2.

Step 4 If $k = 0$ **stop**.
Else, compute the TOP value for period $\widehat{t}_{k,1}+1$ for each job i in $\mathbf{E}_k$.

Step 5 If $\mathbf{E}_k^+ = \{\emptyset\}$ or $\sum_{t=\widehat{t}_{k,0}}^{f_i} \sum_{j\in\mathbf{E}_k} v_{j,t} = 0$ then $k \leftarrow k-1$ and go to Step 4.
Else, $j \leftarrow \arg\max_{i\in\mathbf{E}_k^+} TOP_{\widehat{t}_{k,1}+1-\widehat{d}_i}(i)$; Go to Step 6.

Step 6 Apply the FD algorithm to the current schedule assuming $\widehat{d}_j = \widehat{t}_{k,1}+1$.
If the objective function improves, replace the current schedule with the new one; $\widehat{d}_j \leftarrow \widehat{t}_{k,1}+1$; Go to Step 7.
Otherwise, keep the current schedule; $\mathbf{E}_k^+ \leftarrow \mathbf{E}_k^+ - \{j\}$; Go to Step 5.

Step 7 If $\widehat{t}_{k,1}+1 = \widehat{t}_{k-1,0}$ then a new block is formed; Go to Step 8.
Else, update $\widehat{t}_{k,0}$ and $\widehat{t}_{k,1}$ for LFD block k; Go to Step 4.

Step 8 If $k = M-1$, go to Step 0.
Else, update $\widehat{t}_{k,0}$ and $\widehat{t}_{k,1}$ for LFD block k; Go to Step 4.

In block 1, the TOP values that result with moving the finish date to period 6 return 0, -5, and 1 for jobs 1, 2, and 3 respectively. Consequently tardiness relaxation is attempted only for job 3. Applying the FD algorithm assuming that the due date for job 3 is extended to period 6, we get a new full-delay schedule depicted in Figure 4.8. In the new schedule, the objective function is reduced by 1 unit because the overtime in period 4 is eliminated. Therefore, we adopt the new schedule with the improved objective function value and now, $\widehat{d}_3 = 6$. Observe in the figure that because the regular capacity gap is closed, blocks 1 and 2 are joined to become a single block. Hence, since this block is the only block now, the algorithm restarts with it.

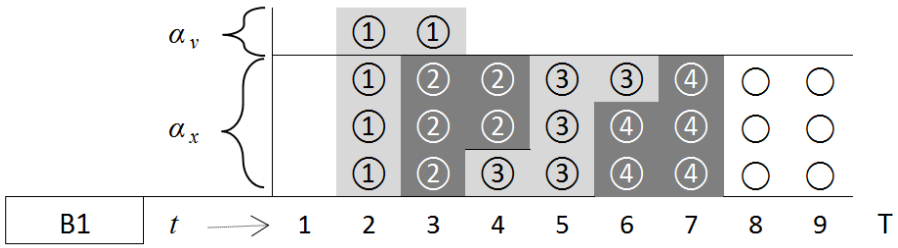

Figure 4.8: Improved schedule after the first pass

Now, the algorithm attempts to improve the objective function by considering shifting the block to the right, no more than one period at each pass. In the new schedule, extending the completion time to period 8 for jobs 1-4 results with TOP values of -3, -10, 2, and -1 respectively. Returning the only positive value, job 3 is again the candidate for the next shift. Assuming now its due date is period 8 and applying the FD algorithm, we get the full-delay schedule in Figure 4.9. The objective function is improved by 2 units with the new schedule. As such, we adopt the new schedule with the improved objective function value and let $\widehat{d}_3 = 8$. Since, overtime usage is eliminated in the new schedule, we stop the process. Clearly, in many instances, overtime usage cannot be completely eliminated. In such cases, the algorithm terminates when no more improvement is gained.

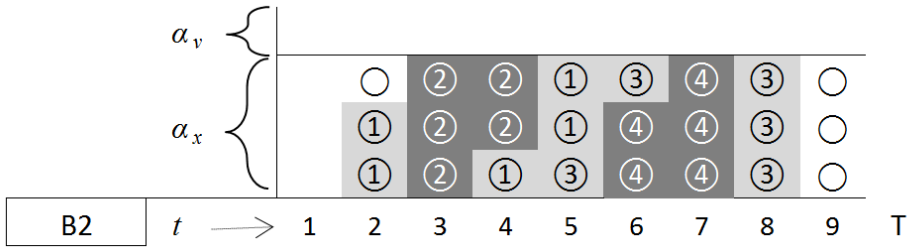

Figure 4.9: Improved schedule after the second pass

4.4.1 Computational Experiments

In order to test the effectiveness of the TWTOT Heuristic via computational experiments, a numerical analysis is designed and implemented. Similar to our computational experiments for the TWT heuristic, we consider seven problem sizes with 7, 10, 15, 20, 30, 40 and 80 jobs resulting with 140 instances. To generate our instances, we employ again the parameter generating expressions in Table 4.2. In addition, we use UNIF$(0.1, a_3)$ to generate costs for overtime, where $a_3 = 0.8\bar{w}$ and $\bar{w}$ represents the average weight (marginal cost of tardiness) over all jobs. For determining the overtime capacity in each instance, we employ UNIF$(0.1, \alpha_R/3)$ if $\alpha_R \geq 1/3$, and UNIF$(0.1, \alpha_R)$ otherwise. This way, we limit the overtime capacity below one third of the regular capacity to be more consistent with real conditions.

As in the previous case, all instances were solved by using a PC with an I7-3770 Processor with 3.40 GHz and 16 GB of RAM. For the exact model, AMPL and CPLEX 12.6.1 were used. Due to the computational complexity of the TWTOT problem, not all the instances are solvable in reasonable amounts of time. Therefore, we maintain a time limit for the execution of the exact model of 7200 seconds (2 hours). Following the same format used for the TWT heuristic, in cases where instances were solved optimally by the exact model we compare the heuristic solutions to the optimal solutions, and in cases where the exact model did not reach optimality, we compare the best feasible solution obtained after two hours with the solution provided by the heuristic. The TWTOT heuristic was implemented using MATLAB R2016a.

4.4.2 Computational Results

We investigate the results of all all 140 instances (20 for each problem size) on each solution approach (280 runs). In our numerical analysis we tracked the number of instances that were solved optimally through each approach within the given time frame. The exact model was able to find the optimal solution in 32% of the instances (45 of the 140 instances) while the heuristic produced 21 optimal solutions.

While the exact model reports optimal solutions for most of the instances of the smaller problem sizes (7, 10 and 15 jobs), no optimal solutions are achieved for the instances with 20, 30, 40 and 80 jobs after two hours. Conversely, and similarly to the TWT heuristic, the TWTOT heuristic was able to reach near optimal solutions (gap $< 5\%$) on 26 out of the 45 optimal instances and a smaller total tardiness cost than the exact model in 74 out of the 95 instances that timed out.

The computational time performances and the percent error resulting from the the heuristic approach as compared with the exact solution are summarized in in tables 4.7 and 4.8 as they relate to the number of instances solved optimally (OPT) and the number of instances that timed out (TO). Measures of computational time performance are presented in seconds and classified by the minimum, average, and maximum time values observed across all problem sizes while the heuristic error is presented via the minimum, average and maximum values of the error relative to the optimal result (for instances solved optimally) and relative to the best solution reached by the exact model (for instances that timed out).

Again, the results indicate a significant difference between the heuristic and the exact model in terms of computational times. In relatively small size problems (*i.e.* 7, 10 and 15 jobs), while it takes on average 6.5 minutes for the exact model to reach

Table 4.7: Optimal Instances

	Exact Model				Heuristic					
	Comp. Time (sec)				Comp. Time (sec)			Percent Error (%)		
Jobs	OPT	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
7	20	0.172	47.88	696.9	0.006	0.016	0.104	0.00	0.93	9.48
10	15	1.17	152.9	1548	0.005	0.014	0.089	0.00	3.51	14.61
15	10	1.248	999.3	3666	0.007	0.008	0.010	0.00	2.36	5.43

Table 4.8: Timed-Out Instances

	Exact Model				Heuristic					
	Comp. Time (sec)				Comp. Time (sec)			Percent Error (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max	Worst	Avg	Best
10	5	7204	7500	8040	0.007	0.010	0.019	2.31	1.26	0.52
15	10	7205	8073	8997	0.007	0.017	0.070	3.30	0.18	-2.48
20	20	7202	7750	8818	0.007	0.010	0.016	8.91	-0.57	-4.09
30	20	7201	7678	8176	0.008	0.013	0.021	10.97	-2.47	-11.6
40	20	7206	7720	8389	0.011	0.018	0.027	0.92	-5.18	-11.1
80	20	7211	7533	8360	0.016	0.048	0.136	-2.94	-17.6	-39.7

optimality, the heuristic achieves the same result or near within a fraction of a second. As the problem size increases, the convergence performance degrades significantly for the exact model. The average optimality gap stays above 35% for more than 20 jobs with a worst case performance of 90.3% gap. On the other hand, similarly to the TWT heuristic as the problem size increases the quality of the heuristic solution shows a relative improvement over the upper bound provided by the exact solution method; that is, the heuristic solution is on average 5.18% and 17.61% better than the best solutions obtained by the timed-out exact model for 40 and 80 jobs respectively. The summary of the relative improvement of the heuristic with respect to the exact

solution is depicted in Figure 4.10. It is clear from the figure that the effectiveness of the proposed heuristic becomes more significant as the problem size grows.

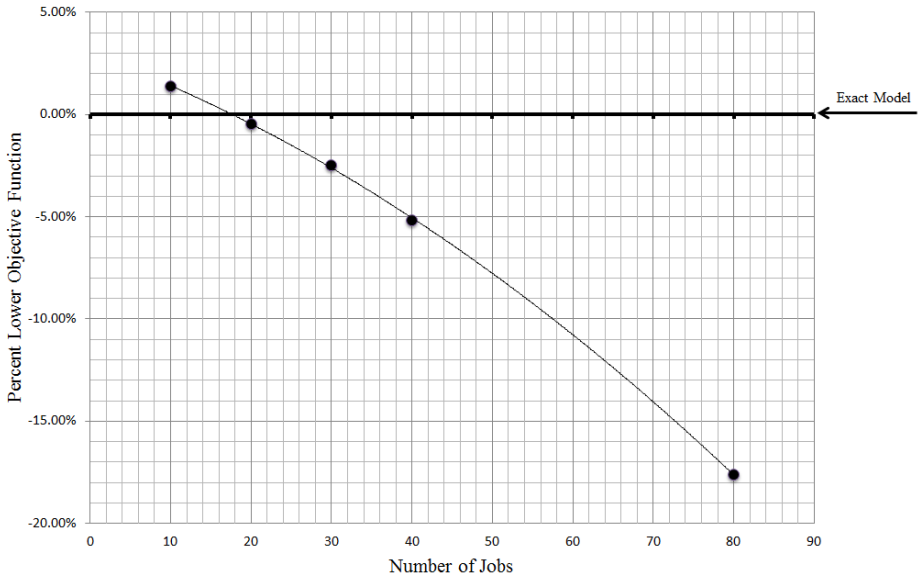

Figure 4.10: TWTOT Heuristic average performance vs. exact model

4.5 Conclusions

This chapter tackles the single machine preemptive scheduling problem that aims to minimize a composite cost function of weighted total tardiness and overtime capacity utilization. The fact that preemption is allowed requires a different methodology compared to its non-preemptive equivalent reported in the literature. In this setting, the workload associated with each job can be allocated across regular and overtime capacities throughout the planning horizon without following strict sequences thanks to the preemption option. We first present a MIP model that can be used to generate optimal schedules for relatively small size problems. Later we propose a heuristic (TWT Heuristic) based on a priority rule for the version of the problem that does

not consider the overtime capacity option. In this method, the priorities of the jobs are determined by their weights (cost of tardiness), due dates, and remaining workload to complete. The heuristic priority rule was tested against the MIP model both in terms of computational performance and quality of the solutions. It was shown that the simple priority rule proposed in this chapter is effective in obtaining efficient schedules. The priority rule can be easily used for generating quick results for large size problems.

A holistic approach (TWTOT Heuristic) is developed to tackle the general version where overtime capacity option is available. The developed method generates a full-delay schedule that allocate workloads among regular and overtime capacity slots based on the critical trade-off between the tardiness and overtime costs. The method is carried out in three stages. In the first stage, the TWT algorithm is applied treating all overtime capacity as regular and hence, ignoring the overtime costs. Second stage reduces the overtime usage by generating a full-delay schedule without altering the tardiness of jobs produced in the first stage (FD Algorithm). Finally, the third stage improves the total costs by extending the tardiness of jobs in return for overtime reduction as long as a net gain is realized (TR Algorithm). The computational tests are repeated for this method and contrasted with the MIP model solutions. The results show that the proposed method can serve as an efficient tool for obtaining good quality solutions in significantly shorter time duration.

An intended extension to our work is to study the multi-resource version of this problem. In this extension, the workloads can be allocated among parallel resources (or work teams) in a given period. This problem may be studied under various settings. For example, the scheduling approach can be influenced by whether the

job allocations are allowed to switch across resources or whether workloads can be simultaneously split to different processors.

CHAPTER 5

Conventional vs. Advanced Models

5.1 Overview

In this chapter, we introduce two mixed integer programming models using two different approaches for the problem of finding a preemptive schedule that minimizes the total cost of tardiness and overtime for a set of jobs with identical processing times on a single processor. The first approach views the problem mainly from the perspective of scheduling (the conventional approach). As such, the ensuing model lends itself to the scheduling paradigm where it aims to identify optimal start and finish times for all jobs. Here we develop a mixed integer programming model with binary scheduling variables, which is referred to as the *Binary Preemptive Scheduling (BPS) Model* .

The second modeling approach (the advanced model) employs the workload planning view; here, existing capacity is distributed among jobs where jobs are aggregated based on their weights as workloads to be fulfilled. In that regard, the approach borrows from the aggregate planning paradigm; this modeling approach eliminates binary variables and provides significant improvement in computational performance. As such, we refer to it as the *Aggregate Preemptive Scheduling (APS) Model*.

We demonstrate with numerical experimentation and analytical methods that the advanced modeling approach (APS) is more efficient and computes better lower bounds than the conventional model; also, we validate such results by using the APS model in a real industry case where we compute well under four seconds the optimal overhaul schedule for 82 jobs under four different customers. Overall findings show that the proposed advanced model is an effective tool for generating optimal or near-optimal schedules reasonably quick for real life applications.

5.2 The BPS Model for TWTOT

The BPS optimization model aims to find an optimal schedule that minimizes the total cost of tardiness and overtime for a given set of jobs by determining regular and overtime work allocations, finish dates, and tardiness. The nomenclature for the decision variables of the model is given in Table 5.1.

Table 5.1: Decision Variables of the BPS Model

x_{it}	Fraction of a job i assigned during regular time at period t
v_{it}	Fraction of a job i assigned during overtime at period t
y_{it}	Binary variable: 1 if job i finishes on period t, 0 otherwise
f_i	Finish date for job i
l_i	Number of late periods on job i

The BPS model involves binary, continuous, and discrete decision variables. The binary variables y_{it} track the periods in which jobs are completed. While the continuous variables determine the amount of allocations of regular capacity x_{it} and overtime work v_{it} that are necessary to complete the jobs, the discrete variables are needed to evaluate the finish date f_i as well as the number of late periods l_i. The tardiness cost

for job i is captured by $w_i l_i = w_i \times max\{0, f_i - d_i\}$. Consequently, we can write down the mathematical model as follows:

$$\text{minimize} \quad \sum_{i=1}^{n} w_i l_i + c_v \sum_{i=1}^{n} \sum_{t=1}^{T} v_{it} \tag{5.1}$$

subject to:

$$\sum_{t=r_i}^{T} y_{it} = 1, \qquad \forall i \in I, \tag{5.2}$$

$$f_i = \sum_{t=1}^{T} t y_{it}, \qquad \forall i \in I, \tag{5.3}$$

$$l_i \geq f_i - d_i, \qquad \forall i \in I, \tag{5.4}$$

$$\sum_{i=1}^{n} x_{it} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\}, \tag{5.5}$$

$$\sum_{i=1}^{n} v_{it} \leq \alpha_v, \qquad \forall t \in \{1, 2, \ldots, T\}, \tag{5.6}$$

$$\sum_{t=r_i}^{T} (x_{it} + v_{it}) = 1, \qquad \forall i \in I, \tag{5.7}$$

$$\sum_{t=1}^{r_i - 1} (x_{it} + v_{it}) = 0, \qquad \forall i \in I, \tag{5.8}$$

$$\sum_{t=r_i}^{\gamma} (x_{it} + v_{it}) \geq y_{i\gamma}, \quad \forall i \in I, \ \forall \gamma \in \{r_i, \ldots, T\}, \tag{5.9}$$

$$x_{it}, \ v_{it} \in [0, 1]\,; \quad f_i, \ l_i \in \mathbb{Z}^+; \quad y_{it} \in \{0, 1\}. \tag{5.10}$$

The objective function given in (5.1) minimizes the total cost of tardiness and overtime capacity. As mentioned above, the weight for job i, w_i, maps the unit cost of tardiness for that job in this representations. The first set of constraints (5.2) ensure that each job is completed by the end of the planning horizon. The second set of constraints (5.3) capture the completion times. Given completion times, constraint in (5.4) sets the tardiness for the jobs. Constraints (5.5) and (5.6) enforce the regular

and overtime capacity limits. Constraints (5.7) and (5.8) allocate all work required to complete 100% of the work on each job across time periods following its release date and (5.9) ensures that jobs are not completed before all required work is done. Solution to the above model provides an optimal preemptive schedule for a given set of jobs with different release and due dates.

The above model is the product of a conventional modeling approach for the scheduling problems. As expected, the complexity introduced by the combination of the binary outcomes adversely affects the computational performance. To improve the computational efficiency, we employ an approach borrowed from the aggregate planning and develop an alternative model that eliminates the binary variables.

5.3 The APS Model for TWTOT

The proposed aggregate preemptive scheduling (APS) model eliminates the binary variables by aggregating all jobs that have the same weights into a single mutually exclusive set. From a practical perspective, each set may represent the work orders of a single customer, where customers carry differing significance for the processor. This is in fact the case in the MRO industry, where the criticality of customers vary depending on the contract requirements, business volume, and long-run relations. We let J represent the set of job clusters (or customers) and M_j is the subset of jobs that are aggregated under cluster j ($j \in J$). We assume that the jobs in set M_j are sequenced in nondecreasing order of their due dates and denote the order of job i in M_j with $\Omega_j(i)$.

As mentioned above, each cluster in set J consists of jobs that have the same weight. For example, for $n = 6$ jobs, if a weight vector is given by $\mathbf{W} = \{2, 1, 5, 2, 1, 1\}$

for jobs 1 through 6 respectively then the first job and fourth job belong to the same cluster. Likewise, second, fifth, and sixth jobs are aggregated into a single cluster, and third job falls in a separate cluster. Thus, the set of clusters becomes $J = \{1, 2, 3\}$ with the new aggregated weight vector $\overline{\mathbf{W}} = \{1, 2, 5\}$ that maps the weights of job clusters 1, 2, and 3 respectively. Also suppose that the due-date vector is $\mathbf{d} = \{6, 7, 6, 8, 7, 9\}$ for jobs 1 to 6 respectively in this above example. Consequently, $M_1 = \{2, 5, 6\}$, $M_2 = \{1, 4\}$, and $M_3 = \{3\}$.

While the BPS model employs a one-dimensional array to specify the due date for each job, the APS model requires two-dimensional arrays that specify a job cluster and the dates on which its jobs are due. We introduce D_{jt} which denotes the total number of jobs belonging to cluster j that are due at time t. In the above example $D_{1,7} = 2$ since cluster 1 has two jobs, namely jobs 2 and 5, that are due at time 7. Thus, $D_{1,9} = D_{2,6} = D_{2,8} = D_{3,6} = 1$, and $D_{jt} = 0$ for all other (j, t) pairs.

We modify the decision variables in accord with the proposed approach (Table 5.2). Similar to BPS model, we measure X_{jt} and V_{jt} as the fraction of work required to complete a single job. We introduce two new variables, namely E_{jt} and B_{jt}, that capture the fraction of the completed work before the due date and the incomplete work past due for cluster j at time t, respectively. These two variables are mapped to the number of late jobs L_{jt}.

Table 5.2: Decision Variables of the APS Model

X_{jt}	Fraction of work assigned to cluster j during regular time at period t
V_{jt}	Fraction of work assigned to cluster j during overtime at period t
E_{jt}	Fraction of completed work by deadline for cluster j in period t
B_{jt}	Fraction of unfulfilled work past due for cluster j in period t
L_{jt}	Number of late jobs in period t corresponding to cluster j

The mathematical model is

$$\text{minimize} \quad \sum_{j\in J}\sum_{t=1}^{T} \overline{w}_j L_{jt} + c_v \sum_{j\in J}\sum_{t=1}^{T} V_{jt} \tag{5.11}$$

subject to:

$$B_{j1} - E_{j1} = D_{j1} - K_{j1}, \qquad \forall j \in J \tag{5.12}$$

$$B_{jt} - E_{jt} = D_{jt} + B_{jt-1} - K_{jt} - E_{jt-1}, \ \forall j; \ t \in \{2,3,.,T\} \tag{5.13}$$

$$K_{jt} = X_{jt} + V_{jt}, \qquad \forall j \in J, \forall t \in \{1,2,\ldots,T\} \tag{5.14}$$

$$\sum_{j\in J} X_{jt} \leq \alpha_x, \qquad \forall t \in \{1,2,\ldots,T\} \tag{5.15}$$

$$\sum_{j\in J} V_{jt} \leq \alpha_v, \qquad \forall t \in \{1,2,\ldots,T\} \tag{5.16}$$

$$\sum_{t=1}^{r_{ji}-1} (X_{jt} + V_{jt}) \leq (\Omega_j(i) - 1), \qquad \forall j \in J; \ i \in M_j \tag{5.17}$$

$$L_{jt} \geq B_{jt}, \qquad \forall j \in J; \ t \in \{1,2,\ldots,T\} \tag{5.18}$$

$$X_{jt}, \ V_{jt}, \ B_{jt}, \ I_{jt} \in \mathbb{R}^+, \qquad \forall j \in J; \ t \in \{1,2,\ldots,T\} \tag{5.19}$$

$$L_{jt} \in \mathbb{Z}^+, \qquad \forall j \in J; \ t \in \{1,2,\ldots,T\} \tag{5.20}$$

In the above model, the first two constraints are the work balance equations. These constraints are needed to capture the fraction of late work for the job clusters. Constraints (5.15) and (5.16) enforce the regular and overtime capacity limits. Constraint (5.17) ensures that no work is assigned for a job before its release date. Constraint (5.18) captures the number of late jobs in job clusters on each period. Similar to the BPS model, solution to the above model provides a feasible optimal preemptive schedule for a given set of jobs with different release and due dates.

The use of aggregation in the APS model takes advantage of the fact that in practical applications job schedules contain multiple jobs that belong to a single customer (class). It does not only eliminate the binary variables but also reduces the

problem size compared to the BPS model. For example, consider a set of 20 jobs that must be scheduled over a time horizon T. Suppose also that, of this 20 jobs, 15 belong only to 3 customers while the remaining 5 jobs belong to 5 other customers. In this case, while the BPS model would generate an array of $20 \times T$ decision variables x_{it} (regular time work assignment), the APS model would only necessitate an array of $8 \times T$ for decision variable X_{jt}. It is expected that the elimination of the binary variable ameliorates the combinatorial nature of the problem by relaxing the solution space. In what follows we demonstrate the computational efficiency of the proposed model via analytic methods and extensive numerical analysis.

5.4 Analytical Comparison of Models

In this part of our discussion we continue the comparison of the presented models this time under a mathematical umbrella. The sections ahead are targeted to examine three different aspects that provide the foundations to the characteristic performance of both models; the first section shows that the BPS as well as the APS model are equivalent models therefore they provide the same solution to the same problem; the second section takes a look at the solution space of both models and shows the differences that make the APS model a faster model. Finally the last section we explore the APS and BPS model structure to explain why the APS model consistently provides better lower boundaries (*i.e* smaller optimallity gaps) than the BPS model.

5.4.1 Model Equivalency

From a mathematical perspective we show that the BPS and APS models are equivalent in terms of the objective functions and constraints and that they both

solve the same Problem. Our approach is a basic one, and lies on showing that both models have equivalent objective functions and equivalent constraints that result in matching optimal solutions.

From evaluating each one of the terms in the objective functions we can see that the tardiness cost variables (l_i and L_{jt}) don not map directly from one model to the other. While in the BPS model l_i represents the actual number of tardy periods of job i, the variable L_{jt} in the APS model only tracks the individual periods in which a job i in customer j is actually late.

$$BPS: \qquad \sum_{i=1}^{n} w_i l_i + c_v \sum_{i=1}^{n} \sum_{t=1}^{T} v_{it}$$

$$APS: \qquad \sum_{j \in J} \sum_{t=1}^{T} \overline{w}_j L_{jt} + c_v \sum_{j \in J} \sum_{t=1}^{T} V_{jt}$$

For both objective functions to match one-to-one, the tardiness expressions $\sum_{t=1}^{T} L_{jt}$ and l_i must be equivalent as shown in expression (5.21).

$$l_i = \sum_{t=1}^{T} L_{jt} \tag{5.21}$$

In order to validate the equivalence in (5.21) the summation of L_{jt} over t must amount to the same tardiness represented by $l_i = f_i - d_i$ in the BPS model; therefore,

$$\sum_{t=1}^{T} L_{jt} = f_i - d_i \tag{5.22}$$

Since $L_{jt} \geq B_{jt}$ as listed in constraints (5.18) of the APS model, it is clear that each tardy period L_{jt} is tied to periods in which there is pending work ($B_{jt} > 0$); furthermore, B_{jt} is constrained to satisfy the work balance equation $B_{jt} - E_{jt} =$

$D_{jt} + B_{jt-1} - X_{jt} - V_{jt} - E_{jt-1}$ where the variables E_{jt} cannot be greater than zero on periods t where $B_{jt} \geq 0$ and E_{jt} cannot be greater than zero on periods $t \geq argmax_{t\in\{1,2,\ldots T\}}(D_{jt})$. Being the precise period when a job is due t_d represented by:

$$t_d = argmax_{t\in\{1,2,\ldots T\}}(D_{jt}) \tag{5.23}$$

and the value of B when the job is due B_{jt_d} is

$$B_{jt_d} = D_{jt_d} - (X_{jt_d} + V_{jt_d} + E_{jt_d-1}) \tag{5.24}$$

Where $B_{jt_d} \in [0, D_{jt_d}]$ and $X_{jt_d}, V_{jt_d} \in [0, D_{jt_d}]$.

In light of the above premises (expressions 5.23, 5.24) we can express the total tardiness of a job i in j as

$$\sum_{t=1}^{T} L_{jt} + t_d - t_d = \sum_{t=1}^{T} \lceil B_{jt} \rceil + t_d - t_d \tag{5.25}$$

Where on the left side $\sum_{t=1}^{T} L_{jt} + t_d$ is equivalent to the finish date which we label as t_f and on the right side can be broken down into the summation of B values before the due period ($t \leq t_d - 1$), between the due period and the finish period ($t_d \leq t \leq t_f - 1$), and after the finish period ($t \geq t_f$)

$$\sum_{t=1}^{T} B_{jt} = \sum_{t=1}^{t_d-1} B_{jt} + \sum_{t=t_d}^{t_f-1} B_{jt} + \sum_{t=t_f}^{T} B_{jt} \tag{5.26}$$

Here the first and last term on the right side are equivalent to zero and the middle term represents the equivalent non-integer expression for tardiness where $\sum_{t=t_d}^{t_f-1} B_{jt} + t_d$ represents the non-integer expression for the finish date, and showing that the amount of tardiness is directly equivalent from one model to another as in (5.27)

$$l_i = \sum_{t=t_d}^{t_f-1} \lceil B_{jt} \rceil + t_d - t_d \tag{5.27}$$

Further inspection of the objective functions in both models reveals that the second terms representing the overtime cost allocation have a full direct correspondence between the models; however, outside of the objective function the regular time and overtime work allocation variables ($x_{it}, X_{jt}, v_{it},$ and V_{jt}) do not map directly from one model to the other; this is, the value of an x_{it} in the BPS model does not necessarily corresponds to the value of X_{jt} in the APS model due to the aggregation of jobs i into the clusters j - similarly for v_{it} and V_{jt}; however, this technical detail does not interfere with the correspondence of the work allocation constraints (5.5, 5.6, and 5.15, 5.16) since in the binary model the sum of different work allocations of overtime capacity v_{it} over the same period t is directly equivalent to the sum of overtime work allocated V_{jt} over the same period. Similarly the total regular time work allocation over a period t corresponds directly to the total regular time work allocated X_{jt} over t. Using this correspondence the following constraints can be confirmed to be equal:

$$\forall t : \sum_{i=1}^{n} x_{it} = \sum_{j \in J} X_{jt}$$

$$\forall t : \sum_{i=1}^{n} v_{it} = \sum_{j \in J} V_{jt}$$

In terms of preventing work allocations before the release date, constraints (5.8) and (5.17) perform an equivalent function in the BPS and APS models respectively; in the BPS model (5.8) explicitly guarantees that no work allocations are done before the release dates r_i while in the APS model constraint (5.17) tracks each job i in

cluster j not to violate the release date by making sure that the total amount of work performed before such a job corresponds to the job order $(\Omega_j(i) - 1)$; this is:

$$\sum_{t=1}^{r_{ji}-1} (X_{jt} + V_{jt}) \leq (\Omega_j(i) - 1) \qquad \forall j \in J;\ i \in M_j \tag{5.28}$$

Thus far we have shown how the BPS and APS models are equivalent in terms of the work allocation and objective functions; however, the different modeling approaches used imply that the arrangements of constraints do not necessarily map one-to-one across models. Such is the case of the correspondence between the work balance equations in the APS model and the following constraint (5.9) in the BPS model:

$$\sum_{t=r_i}^{\gamma} (x_{it} + v_{it}) \geq y_{i\gamma}, \qquad \forall i \in I,\ \forall \gamma \in \{r_i, \ldots, T\} \tag{5.29}$$

As it is the case, each model resorts to a different approach to guarantee that the work allocated does not violate the allocation limits set by the release and finish dates. In the case of the APS model the work balance equations automatically guarantee that the allocated work fulfills all the jobs demand by the time the job is finished. In the case of the BPS model, due to the lack of work balance equations and the fact that the variables y_{it}, x_{it}, and v_{it} are totally independent, constraint (5.29) above is required to serve as coupling between the work performed tracked by x_{it}, and v_{it} and the finish date y_{it}. Mathematically, constraint (5.29) represents a breakdown of constraint (5.7) where γ controls the different subsets of periods in between $[r_i, T]$ that must be satisfied. Given that $\gamma \in \{r_i, r_i + 1, r_i + 2, \ldots, T\}$ there is a γ such that $\gamma = t_f$ and $y_{it_f} = 1$. In this particular period, constraints (5.29, must be satisfied as follows

$$\sum_{t=r_i}^{t_f}(x_{it}+v_{it}) \geq 1$$
$$\sum_{t=r_i}^{T}(x_{it}+v_{it}) = 1$$

forcing constraint 5.29 to bind the binary value of 1. and the interval complements $\sum_{t=1}^{r_i-1}(x_{it}+v_{it})$ and $\sum_{t=t_f+1}^{T}(x_{it}+v_{it})$ must be equal to zero therefore performing an analogous function to work balance equations in the APS model.

This analysis shows that both models solve the same problem not only by having equivalent objective functions but also by having sets of constraints that define an equivalent solution space across both models.

5.4.2 Solution Space and Model Performance

In this section we use the solution space as an indicator of the computational burden in each model; in particular, we show why the APS model possesses a lower computational burden than the BPS model which results in faster solutions. We initiate this discussion using the BPS model to computing the size of its solution space; this is, the number of possible combinations of independent variables required to generate a single feasible solution. In this particular case a single feasible solution with a cost

$$\psi = \sum_{i=1}^{n} w_i l_i + c_v \sum_{i=1}^{n}\sum_{t=1}^{T} v_{it} \tag{5.30}$$

Being T the total number of periods in the planning horizon and n the total number of jobs in the set of jobs I; the model must compute the lateness for each one

of the n jobs (*i.e.* $l_i : i \in \{1, 2, \ldots, n\}$) where $l_i = f_i - d_i$ and $f_i = \sum_{t=1}^{T} t y_{it}$ subject to $\sum_{t=1}^{T} y_{it} = 1, \forall i \in I$. Considering this, to produce a single feasible value for l_i the independent binary variable y_{it} must try from a combination of binary options in the elements of a vector $\hat{t} =< a_1, a_2, \ldots, a_T >$ where each element $a_t \in \{0, 1\}$ for $t \in \{1, 2, \ldots, T\}$. Given T binary digits in each job there are only T feasible options for y_{it} to take the value of 1 only once within $\hat{t}$; for an instance, if $T = 3$ there will only be three possible options for y_{it} to take the value of 1. this is illustrated as follows in figure 5.1

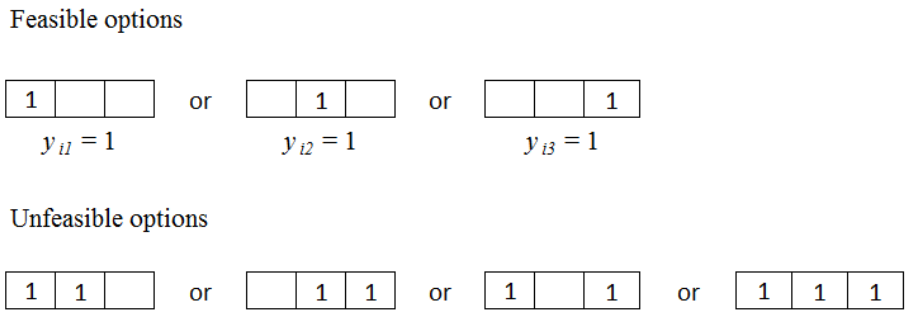

Figure 5.1: Sample feasible and unfeasible options for y_{it}

Here the total number of possible non-zero value options S_T (feasible and unfeasible) for an independent variable like y_{it} on a single job is:

$$S_T = T + \frac{T!}{2!(T-2)!} + \frac{T!}{3!(T-3)!} + \cdots + \frac{T!}{T!(T-T)!}$$

$$S_T = \sum_{k=1}^{T} \frac{T!}{k!(T-k)!}$$

$$S_T = \sum_{k=1}^{T} \binom{T}{k}$$

In addition to the number of options S_T for y_{it}, variables x_{it}, v_{it} are also independent, and there are also S_T possible non-zero value options for each one of these. Considering that for a single feasible solution the solution space created by x_{it}, y_{it}, v_{it} is determined by S_T number of options we have that for a single job the solution sub-space in the BPS model SS_{job} is given by :

$$SS_{job} = S_T \times S_T \times S_T = (S_T)^3 \tag{5.31}$$

and for a group of n jobs the BPS model solution space SS_{BPS} is:

$$SS_{BPS} = n(S_T)^3 \tag{5.32}$$

Analogously, from the variables in the APS model it can be noted that only X_{jt}, V_{jt} are independent. Following the same logic from the binary model the number of non-zero value options for X_{jt} and V_{jt} is also given by S_T, and since there are only two independent variables in the APS model we can conclude that the total solution space for the APS model is given by SS_{APS}

$$SS_{APS} = n(S_T)^2 \tag{5.33}$$

which indicates that the solution space for the BPS model is S_T times larger than the one for the APS; furthermore, we can conclude that since the variables X_{jt} and V_{jt} map directly to the variables x_{it} and v_{it} in the BPS model, the APS solution space is contained in the BPS solution space:

$$SS_{APS} \subset SS_{BPS} \qquad \square \tag{5.34}$$

5.4.3 Lower Boundaries and Model Accuracy

The expected difference in the performance between the models is primarily the result of each models ability to produce a lower bound. In this section we show that the APS model can produce equal or better lower bounds than the BPS model. We start the discussion by defining the lower bound for each model using the definition of an integer relaxation; for this we use LB_{BPS}, and LB_{APS} to designate the BPS and APS model lower bounds respectively with relaxed variables $\hat{y}$ and $\hat{L}$

$$LB_{BPS} = h(\hat{y}): \qquad \hat{y} \in (0,1)$$
$$LB_{APS} = g(\hat{L}): \qquad \hat{L} \in \mathbb{R}^+$$

and their corresponding objective functions

$$LB_{BPS} = \sum_{i=1}^{n} w_i \hat{l}_i + c_v \sum_{i=1}^{n} \sum_{t=1}^{T} v_{it} \qquad where\ \hat{l}_i = f(\hat{y})$$

$$LB_{APS} = \sum_{j \in J} \sum_{t=1}^{T} \overline{w}_j \hat{L}_{jt} + c_v \sum_{j \in J} \sum_{t=1}^{T} V_{jt}$$

We can see that the second terms on both expressions are equivalent to each other and do not depend on the relaxation of any discrete variable; therefore, any difference between the LB_{BPS}, and LB_{APS} does not result from any of these terms and can be ignored in our analysis. Both of the model definitions show that the effect of a linear relaxation would be determined by the relaxation of y_{it} in the case of the BPS model and by L_{jt} in the case of the APS model. In the particular case of the BPS model, the relaxed variables impact the objective function as follows:

$$\hat{y}_{it} = \quad y_{it} : y_{it} \in (0, 1) \tag{5.35}$$

$$\hat{f}_i = \qquad \textstyle\sum_{t=1}^{T} t\hat{y}_{it} \tag{5.36}$$

$$\hat{l}_i = \quad \textstyle\sum_{t=1}^{T} t\hat{y}_{it} - d_i \tag{5.37}$$

Resulting in the lower bound below

$$LB_{BPS} = \sum_{i=1}^{n} w_i (\sum_{t=1}^{T} t\hat{y}_{it} - d_i) \tag{5.38}$$

For the APS model the relaxation of the integer variable L_{jt} implies that the lateness on an individual period needs to bind the variable B_{jt} as follows

$$\hat{L_{jt}} = B_{jt} \qquad \forall j \in J, \text{ and } \forall\ t \in \{1, 2, \ldots, T\}$$

Resulting in the following bound

$$LB_{APS} = \sum_{j \in J} \sum_{t=1}^{T} \overline{w}_j B_{jt}$$

which can also be expressed in terms of the finish date using the due date d_{ji} as follows

$$LB_{APS} = \sum_{j \in J} \overline{w}_j \left(\sum_{t=1}^{T} B_{jt} + \sum_{i \in M_j} d_{ji} - \sum_{i \in M_j} d_{ji} \right) \tag{5.39}$$

From evaluation of the lower boundaries from each model (Expressions 5.38, and 5.39) it can be seen that their different values are determined by how the relaxed finish dates are computed in each model (f_{APS}, f_{BPS}); therefore, showing that the relaxed finish date for any job in the APS model is larger or equal than the corresponding relaxed finish date calculated by the BPS model suffices to show that $LB_{BPS} \leq LB_{APS}$. To this end we focus our discussion on the expressions that determine the value of the relaxed finish dates for a single given job i, and since clustering is not relevant in the context of an individual job we present the expressions for the relaxed finish dates without regard to the j indices representing each cluster.

$$f_{BPS} = \sum_{t=1}^{T} t\hat{y}_{it}$$

$$f_{APS} = \sum_{t=1}^{T} B_{it} + d_i$$

Before breaking down each one of the relaxed finish dates above, we define the context and notation that will be used through the remaining of this section. In terms of the context or the situation used to frame our reasoning, we know that for any given a job there are three possible schedule scenarios depending on the relationship between the start s_i and finish dates f_i with respect to the due date d_i; listed as follows:

Scenario 1: The job is early $s_i < f_i \leq d_i$

Scenario 2: The job is partially late $s_i \leq d_i < f_i$

Scenario 3: The job is completely late $d_i < s_i < f_i$

For the sake of simplicity we start by showing the difference between the finish dates above using a single job argument under Scenario 3 where preemption is ignored, and then conclude our discussion by showing that any of the other scenarios with or without preemption will produce a larger difference between the relaxed finish dates computed by each model. We use the third scenario as the context of our mathematical argument as this scenario represents the schedule conditions in which the relaxed finish dates (f_{APS}, f_{BPS}) are forced to be in close proximity to each other (*i.e* they tend to the limiting case $f_{APS} = f_{BPS}$).

Let us start the discussion focusing on the BPS model. When a single job is started under the third scenario in the BPS model, full allocation takes place in every period t so as to complete the total amount work to finish each job p_w as soon as possible (in this particular case $p_w = 1$ for all jobs); this means that the regular time work allocation is $x_{it} = \alpha_x$ and the relaxed allocation of $\hat{y}_{it}$ comply with constraint

(5.9) in the model, and the minimization objective in the model forces the value of $\hat{y}_{it}$ to bind the allocated amounts of regular time work as follows:

$$\hat{y}_{i\gamma} = \sum_{t=r_i}^{\gamma} x_{it} \qquad \forall \gamma \in \{r_i, \ldots, T\} \tag{5.40}$$

The allocation of values for $\hat{y}_{it}$ continues in every period t until the model constraint (5.2) is satisfied and reaches the value of 1. This process of allocation is illustrated in figure 5.2

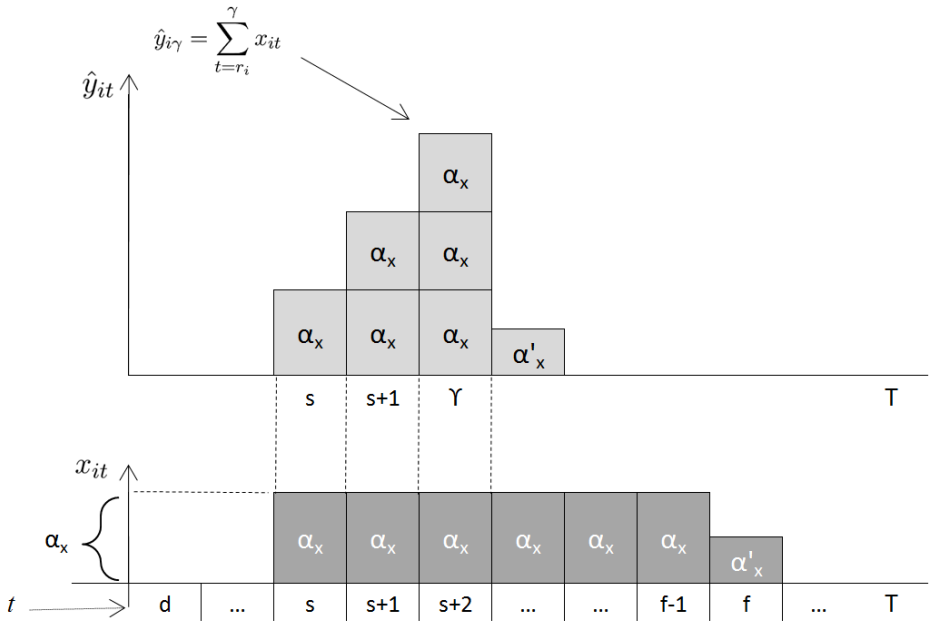

Figure 5.2: Allocation of values for $\hat{y}_{it}$. $(p_w = 1)$

The finish date produced by the BPS model under relaxed conditions can be summarized in expression 5.41.

$$f_{BPS} = \underbrace{\left(\frac{\alpha_x}{p_w}\right) s + 2\left(\frac{\alpha_x}{p_w}\right)(s+1) + \cdots + n_k\left(\frac{\alpha_x}{p_w}\right)(s+n_k-1)}_{A} + \underbrace{n_o\left(\frac{\alpha_x}{p_w}\right)(s+n_k)}_{B} \tag{5.41}$$

The braces labeled (A) and (B) indicate two distinctive groups of expressions in f_{BPS}. On one side, expression (A) represents all the terms whose coefficients fit the characteristics of an arithmetic progression ($\{a_n\}_{n=1}^{n_k} = \{1, 2, 3, \ldots, n_{k-1}, n_k\}$), while expression (B) indicates the last term in f_{BPS} for which n_o does not necessarily follows the integrality requirements of the arithmetic progression formed by the rest of the terms in A. This is, $n_o \leq n_k + 1$. Also expression (B) can be interpreted as the amount necessary $(\alpha_x^{'})$ for $\hat{y}_{it}$ to satisfy the constraint that guarantees that each job is finished $\sum_{t=r_i}^{T} \hat{y}_{it} = 1$; however since this constraint needs to be satisfied the value of n_o is given by the sum of the elements in the arithmetic progression formed by the s-terms present in equation (5.41); this is

$$\sum_{t=r_i}^{T} \hat{y}_{it} = \left(\frac{\alpha_x}{p_w}\right) \frac{n(n+1)}{2} \tag{5.42}$$

where in order to satisfy BPS constraint (5.2) the expression in (5.42) must be equal to 1 as in (5.43)

$$\left(\frac{\alpha_x}{p_w}\right) \frac{n_o(n_o+1)}{2} = 1 \tag{5.43}$$

Solving for n_o produces the value of n_o that satisfies constraint (5.2) where $\lceil n_o \rceil = n_k + 1$

$$n_o = \frac{\sqrt{\left(\frac{\alpha_x}{p_w}\right)^2 + 8\left(\frac{\alpha_x}{p_w}\right)} - \left(\frac{\alpha_x}{p_w}\right)}{2\left(\frac{\alpha_x}{p_w}\right)} \tag{5.44}$$

Using expression (5.43) and rewriting it in terms of n_k and n_o we obtain

$$\left(\frac{\alpha_x}{p_w}\right) n_o + \left(\frac{\alpha_x}{p_w}\right) \sum_{n=1}^{n_k} n = 1 \tag{5.45}$$

Knowing the coefficients in (A) that follow the arithmetic progression $\{a_n\}_{n=1}^{n_k}$ it can be condensed into their corresponding partial series $\sum_{n=1}^{n_k} a_n = \sum_{n=1}^{n_k} n = n_k(n_k+1)/2$ we can use equation (5.45) to solve for $\left(\frac{\alpha_x}{p_w}\right) n_o$ and obtain a new expression for (B) in f_{BPS} in terms of n_k

$$B = \left[1 - \left(\frac{\alpha_x}{p_w}\right) \sum_{n=1}^{n_k} n\right] (s + n_k) \tag{5.46}$$

Reworking the terms in the group of expressions (A) by factoring $\left(\frac{\alpha_x}{p_w}\right)$ and using the series formed by the progression of coefficients we have:

$$A = \left(\frac{\alpha_x}{p_w}\right) [s + 2(s+1) + 3(s+2) + \cdots + n_k(s + n_k - 1)] \tag{5.47}$$

Then expanding and regrouping the elements alike in the progression of coefficients

$$A = \left(\frac{\alpha_x}{p_w}\right) \left[\underbrace{s + 2s + 3s + \cdots + n_k s}_{C} + \underbrace{1(0) + 2(1) + 3(2) + \cdots + n_k(n_k - 1)}_{D}\right] \tag{5.48}$$

where the groups of terms (C) and (D) can be condensed and expressed as

$$A = \left(\frac{\alpha_x}{p_w}\right) \left[\underbrace{s \sum_{n=1}^{n_k} n}_{C} + \underbrace{\sum_{n=1}^{n_k} n(n-1)}_{D}\right] \tag{5.49}$$

here summation (D) can be expanded into $\sum_{n=1}^{n_k} n^2 - \sum_{n=1}^{n_k} n$ resulting into

$$A = \left(\frac{\alpha_x}{p_w}\right) \left[s \sum_{n=1}^{n_k} n + \sum_{n=1}^{n_k} n^2 - \sum_{n=1}^{n_k} n\right] \tag{5.50}$$

The term $\sum_{n=1}^{n_k} n^2$ can be expressed in terms of its formula $n_k(n_k+1)(2n_k+1)/6$ which in turn can be rewritten in terms of $\sum_{n=1}^{n_k} n$ as

$$A = \left(\frac{\alpha_x}{p_w}\right)\left[s\sum_{n=1}^{n_k} n + \left(\frac{(2n_k+1)}{3}\right)\sum_{n=1}^{n_k} n - \sum_{n=1}^{n_k} n\right] \tag{5.51}$$

factoring $\sum_{n=1}^{n_k} n$ we get

$$A = \left(\frac{\alpha_x}{p_w}\sum_{n=1}^{n_k} n\right)\left[s + \left(\frac{2n_k+1}{3}\right) - 1\right] \tag{5.52}$$

Using the expanded expressions for (A) and(B) to work out an expression for $f_{BPS} = A + B$ we have

$$A = \quad s\left(\tfrac{\alpha_x}{p_w}\right)\textstyle\sum_{n=1}^{n_k} n + \left(\tfrac{\alpha_x}{p_w}\right)\sum_{n=1}^{n_k} n\left(\tfrac{2n_k+1}{3}\right) - \left(\tfrac{\alpha_x}{p_w}\right)\sum_{n=1}^{n_k} n \tag{5.53}$$

$$B = \qquad -s\left(\tfrac{\alpha_x}{p_w}\right)\textstyle\sum_{n=1}^{n_k} n + s + n_k - n_k\left(\tfrac{\alpha_x}{p_w}\right)\sum_{n=1}^{n_k} n \tag{5.54}$$

Notice that the first terms in expressions (A) and (B) cancel each other in $f_{BPS} = A + B$ and the coefficients $\left(\frac{\alpha_x}{p_w}\right)\sum_{n=1}^{n_k} n$ can be factored to produce a more compact expression for f_{BPS}

$$f_{BPS} = s + n_k + \left(\frac{\alpha_x}{p_w}\sum_{n=1}^{n_k} n\right)\left[\left(\frac{2n_k+1}{3}\right) - 1 - n_k\right] \tag{5.55}$$

after working on the expressions within the brackets in (5.55) we obtain

$$f_{BPS} = s + n_k - \left(\frac{\alpha_x}{p_w}\sum_{n=1}^{n_k} n\right)\left[\frac{n_k+2}{3}\right] \tag{5.56}$$

which can be expressed in terms of $\lceil n_o \rceil$ by using the equation $\lceil n_o \rceil = n_k + 1$

$$f_{BPS} = s - 1 + \lceil n_o \rceil - \left(\frac{\alpha_x}{p_w}\sum_{n=1}^{\lceil n_o \rceil - 1} n\right)\left(\frac{\lceil n_o \rceil + 1}{3}\right) \tag{5.57}$$

Using the formula to calculate the progression of n numbers applied to $\lceil n_o \rceil - 1$ we convert the summation term in (5.57) to produce

$$f_{BPS} = s - 1 + \lceil n_o \rceil - \left(\frac{\alpha_x}{p_w}\right)\left(\frac{\lceil n_o \rceil(\lceil n_o \rceil - 1)}{2}\right)\left(\frac{\lceil n_o \rceil + 1}{3}\right) \tag{5.58}$$

which after grouping the expressions within the last two parenthesis in (5.58) produce a final expression for f_{BIN}

$$f_{BPS} = s - 1 + \lceil n_o \rceil - \left(\frac{\alpha_x}{p_w}\right)\left(\frac{\lceil n_o \rceil^3 - \lceil n_o \rceil}{6}\right) \tag{5.59}$$

With a compact expression for f_{BPS} now we move to the APS model; in this case, when a single job is started under the third scenario, again full allocation takes place in every period so as to complete the total work p_w as soon as possible; therefore, the regular time work allocation is $x_{it} = \alpha_x$ and the relaxed allocation of $\hat{L}_{it}$, as we showed earlier becomes B_{it}. Under the third scenario the required pending work B_{it} remains constant (*i.e.* equal to 1) after the due date until regular time work is allocated x_{it}, and the amount of pending work decreases until the job is finished. This process is illustrated in figure 5.3.

Analyzing the expression $f_{APS} = \sum_{t=1}^{T} B_{it} + d_i$ we see that the term $\sum_{t=1}^{T} B_{it}$ in a late job is composed of a constant region and a variable (decreasing) region denoted by the superscripts $Con.$ and $Var.$ respectively; this is $\sum_{t=1}^{T} B_{it} = \sum_{t=1}^{s-1} B_{it}^{Con.} + \sum_{t=s}^{T} B_{it}^{Var.}$. The constant part is the part from periods $t = [d, s-1]$ which amounts for $\sum_{t=1}^{s-1} B_{it}^{Con.} = 1 \times (s_i - d_i)$ while the variable part is in periods $t = [s, f-1]$; this leads to a new expression for f_{APS} in terms of s

$$f_{APS} = \sum_{t=1}^{T} B_{it}^{Var.} + s_i \tag{5.60}$$

where $\sum_{t=1}^{T} B_{it}^{Var.}$ can be expanded as follows

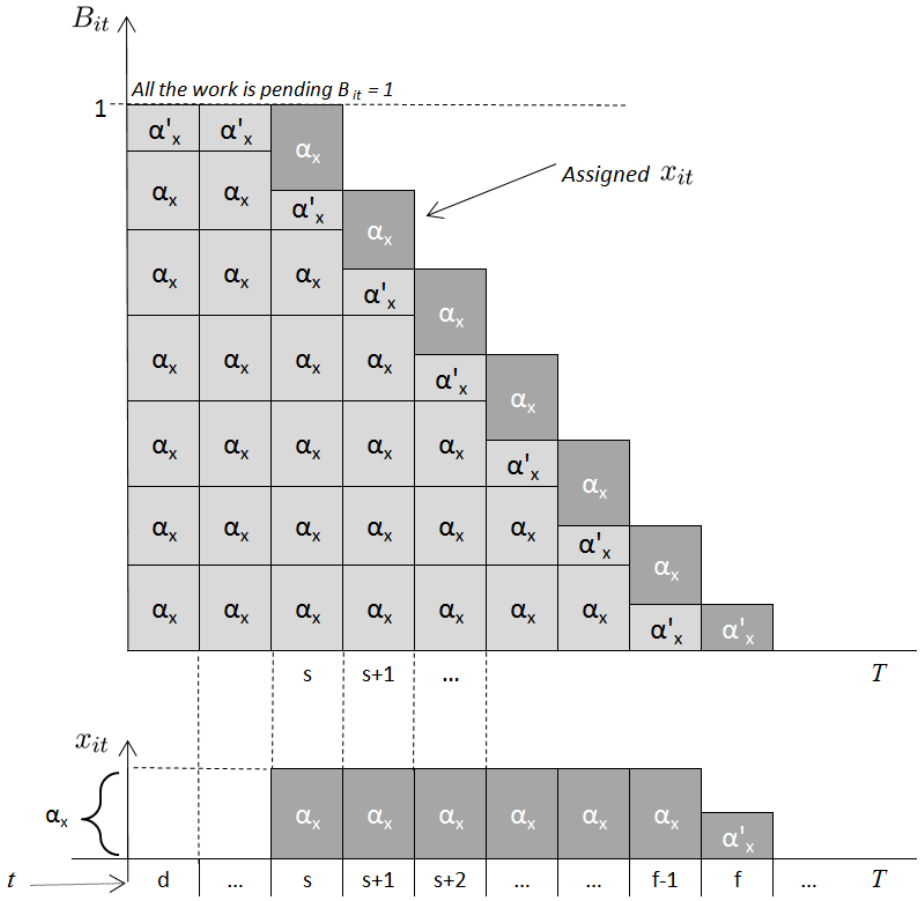

Figure 5.3: Allocation of values for B_{it}. ($p_w = 1$)

$$f_{APS} = s + \left[1 - 1\left(\frac{\alpha_x}{p_w}\right)\right] + \left[1 - 2\left(\frac{\alpha_x}{p_w}\right)\right] + \cdots + \left[1 - \left(\left\lceil\frac{p_w}{\alpha_x}\right\rceil - 1\right)\left(\frac{\alpha_x}{p_w}\right)\right] \quad (5.61)$$

where $\lceil\frac{p_w}{\alpha_x}\rceil - 1$ represents the number of periods from $t = [s, f-1]$ corresponding to $\sum_{t=1}^{T} B_{it}^{Var.}$. Regrouping similar terms and applying the summation formula to the arithmetic progression formed by the coefficients $\{1, 2, 3, \ldots, \left(\lceil\frac{p_w}{\alpha_x}\rceil - 1\right)\}$ we obtain

$$f_{APS} = s + \left(\left\lceil\frac{p_w}{\alpha_x}\right\rceil - 1\right) - \left(\frac{\alpha_x}{p_w}\right)\frac{\left(\left\lceil\frac{p_w}{\alpha_x}\right\rceil - 1\right)\left\lceil\frac{p_w}{\alpha_x}\right\rceil}{2} \quad (5.62)$$

which can be rearranged as

$$f_{APS} = (s-1) + \left\lceil \frac{p_w}{\alpha_x} \right\rceil - \left(\frac{\alpha_x}{p_w}\right) \frac{\left(\left\lceil \frac{p_w}{\alpha_x} \right\rceil^2 - \left\lceil \frac{p_w}{\alpha_x} \right\rceil \right)}{2} \tag{5.63}$$

Now with expressions for the relaxed finish dates on each model f_{APS} and f_{BPS} we can evaluate if the difference $f_{APS} - f_{BPS} \geq 0$ is satisfied in order to prove that the lower bound generated by the APS model is generally larger (better) than the one generated by the BPS model. To this end we evaluate the difference and subtract the finish date on (5.59) from (5.63)

$$f_{APS} - f_{BPS} = \left\lceil \frac{p_w}{\alpha_x} \right\rceil - \left(\frac{\alpha_x}{p_w}\right) \frac{\left(\left\lceil \frac{p_w}{\alpha_x} \right\rceil^2 - \left\lceil \frac{p_w}{\alpha_x} \right\rceil \right)}{2} - \lceil n_o \rceil + \left(\frac{\alpha_x}{p_w}\right) \left(\frac{\lceil n_o \rceil^3 - \lceil n_o \rceil}{6} \right) \tag{5.64}$$

where $\lceil n_o \rceil$ can be expressed in terms of $\left(\frac{p_w}{\alpha_x}\right)$ using equation (5.44); for this we introduce a new variable Z to represent $Z = \left(\frac{p_w}{\alpha_x}\right)$ such that $Z \geq 1$ in order to restrict the relationship between p_w and α_x and not allow the processing of jobs in less than a single time period.

$$\lceil n_o \rceil = \left\lceil \frac{\sqrt{\left(\frac{1}{Z}\right)^2 + 8\left(\frac{1}{Z}\right)} - \left(\frac{1}{Z}\right)}{2\left(\frac{1}{Z}\right)} \right\rceil \tag{5.65}$$

rearranging equation (5.65) we obtain

$$\lceil n_o \rceil = \left\lceil \frac{\sqrt{1+8Z} - 1}{2} \right\rceil \tag{5.66}$$

rewriting $f_{APS} - f_{BPS}$ in terms of $\lceil Z \rceil$ and $\lceil n_o \rceil = f(Z)$ we have

$$f_{APS} - f_{BPS} = \lceil Z \rceil - \lceil n_o \rceil + \left(\frac{1}{Z}\right)\left(\frac{\lceil n_o \rceil^3 - \lceil n_o \rceil}{6}\right) - \left(\frac{1}{Z}\right)\left(\frac{(\lceil Z \rceil^2 - \lceil Z \rceil)}{2}\right) \quad (5.67)$$

Analysis of the terms $\lceil Z \rceil$ and $\lceil n_o \rceil$ show that $\lceil Z \rceil = \lceil n_o \rceil \quad \forall Z \in [0, 2]$; we use this property to restrict the analysis of $f_{APS} - f_{BPS}$ to the range $Z \in [0, 2]$ by substituting $\lceil n_o \rceil$ for $\lceil Z \rceil$

$$f_{APS} - f_{BPS} = \lceil Z \rceil - \lceil Z \rceil + \left(\frac{1}{Z}\right)\left(\frac{\lceil Z \rceil^3 - \lceil Z \rceil}{6}\right) - \left(\frac{1}{Z}\right)\left(\frac{(\lceil Z \rceil^2 - \lceil Z \rceil)}{2}\right) \quad (5.68)$$

rearranging equation (5.68) we obtain

$$f_{APS} - f_{BPS} = \left(\frac{1}{Z}\right)\left(\frac{\lceil Z \rceil^3 - 3\lceil Z \rceil^2 + 2\lceil Z \rceil}{6}\right) \quad (5.69)$$

From which the coefficient $\left(\frac{1}{Z}\right) > 0$ and the polynomial $\lceil Z \rceil^3 - 3\lceil Z \rceil^2 + 2\lceil Z \rceil$ has roots $\{0, 1, 2\}$, is positive for $Z > 2$, and displays upwards concavity for values $Z \geq 1$; al this indicating that $f_{APS} - f_{BPS} = 0 \quad \forall Z \in (1, 2]$ and $f_{APS} - f_{BPS} \geq 0 \quad \forall Z \geq 2$. This shows that the lower boundary for the APS model is equal or greater than the one for the BPS model. $\square$

Keep in mind that up to this point we showed that the lower boundary for the APS model is equal or greater than the one for the BPS model, but we have done so only under the context of Scenario 3 and under no preemption; to complete our discussion on the lower boundaries and extend the results obtained so far to cover Scenarios 1 and 2 under preemption we present the following mathematical arguments.

Proposition 4 *Under preemtion, the relaxed finish dates of a given job (f_{APS} and f_{BPS}) maintain the relationship $f_{APS} - f_{BPS} \geq 0$.*

Proof: From the definition of preemtion we know that any preempted job i will be forced to increase the number of periods by δ_i between the start s_i and finish dates f_i; therefore introducing a delay equivalent to δ_i on the finish dates f_{APS} and f_{BPS} does not change the relationship between the relaxed finish dates as follows where the value of δ_i is not relevant to the inequality $f_{APS} + \delta_i \geq f_{BPS} + \delta_i$. ■

Proposition 5 *the relationship $f_{APS} - f_{BPS} \geq 0$ is also maintained when the due date d_i is located between the start s_i and finish dates f_i (Scenario 2) and when the due date d_i is in a period equal or later than the finish date f_i (Scenario 1).*

Proof: From the expressions representing f_{APS} and f_{BPS}

$$f_{BPS} = \sum_{t=1}^{T} t\hat{y}_{it}$$

$$f_{APS} = \sum_{t=1}^{T} B_{it} + d_i$$

we can see that the relaxed finish date f_{APS} cannot take values smaller than d_i while the value of f_{APS} is independent of the due date. Under the conditions present in Scenario 2 ($s_i \leq d_i < f_i$) the relaxed finish date on the APS model will result on a value equal or larger than d_i while f_{BPS} remains unchanged therefore maintaining the relationship $f_{APS} - f_{BPS} \geq 0$ and biasing it towards producing values of f_{APS} relatively larger than f_{BPS}; furthermore, since f_{APS} cannot take values smaller than d_i under the conditions present in scenario 1 ($s_i < f_i \leq d_i$) the value of f_{APS} will always be larger than f_{BPS} therefore producing a consistent inequality between the relaxed finish dates ($f_{APS} - f_{BPS} > 0$). ■

5.5 Computational Experiments

We design and perform numerical analysis so as to evaluate the computational performance of the BPS and the APS models. We aim to determine the computational performance of both models with respect to the number of jobs, the model parameters, and the job aggregation (clustering) level. Our goal is to demonstrate that the proposed APS model significantly improves the computational performance. To this end we perform three sets of numerical experiments that test both models under different conditions using the number of optimal solutions, computational time and optimality gap as reference measures. In our first numerical experiment, we include in our analysis a varying number of jobs and the aggregation levels. Specifically, we consider five problem sizes with 10, 20, 40, 80, and 160 jobs and test them under different levels of aggregation classified into four groups. In the first group, all jobs are assigned unequal weights and as such aggregation is not possible; in this group both models are tested side by side; In the second group, the same instances from group 1 are used but assigned different weights to produce a different aggregation - in this case at most two jobs can have the same weights and as such, each job cluster (*e.g.*, customer) has at most two members. Similarly, in groups 3 and 4, the same instances are used under different aggregation to make sure each job cluster includes 3 to 5 and 5 to 8 jobs respectively. Since the same instances used in group 1 are adapted to generate groups 2 through 4, and since the BPS model does not use aggregation, the BPS model is not run again in groups 2 through 4 to avoid redundancy in the data reported. Overall the first set of experiments involve 20 different scenarios.

Following the first numerical tests, we proceed with numerical experiments focused only on the APS model. in particular, we use numerical tests to explore how the

performance of the APS model changes depending on how the release and due dates are arranged in the schedule; with this aim we produced four scenarios in which we test the combinations between two levels of release dates (i.e. Condensed Release Dates and Sparse Release Dates) and two levels of due dates determined by the slack added to the jobs (i.e. Constant Slack and Proportional Slack). The four resulting scenarios are tested on the APS model under no aggregation for 20 and 40 jobs. The decision to evaluate the APS model only under the arrangement of release and due dates, on one side, follows observations made by [33] and [24] on the proximity and arrangement of the release and due dates on the complexity of the instance, and on the other, is mostly the result of multiple observations made over extensive testing of the effect of different schedule parameters on the performance of the models; The effect of other parameters like α_x, and T are implicitly tested under the different job levels, and other parameters like α_v and w_i do not demand additional testing as their relationship to the complexity of the instances is more intuitively understood.

Finally, the third set of numerical tests focuses on the special case of the generalized problem $1|pmtn, r_i, p_i = p|\sum_i w_i l_i + c_v \sum_{i,t} v_{it}$ where the overtime capacity is eliminated ($i.e\ \alpha_v = 0$ and the general problem reduces to the TWT problem. In this particular set of tests we use four problem sizes with 10, 20, 30, and 40 jobs and test them under only two different levels of aggregation corresponding to groups 1 and group 4 on the first set of numerical tests. Again, we aim to evaluate the performance of both models under such particular conditions. For all the numerical experiments the lower bound from the APS model is used as reference to produce the optimality gaps in both models; this is necessary due to the tendency of the BPS model to generate artificially small lower boundaries even when the reached feasible solution

is actually close to the optimal. This is discussed in more detail in the analytical comparison of the models Section 5.4.3.

To produce the three phases of numerical experiments described above we use a set of expressions to generate all the parameters necessary to create the problem instances. To be consistent with the existing literature, we borrowed the expressions from earlier research [20] and modified them to our context. Main random variables in the system are generated by a uniform distribution in order to enable diversity across all instances. To avoid trivial cases, the number of jobs n and the maximum allowed regular time capacity α_x are preset and incorporated as input parameters in the problem instance generation process. Table 5.3 lists the specific parameter generating expressions used in our analysis. For each scenario combination, 20 instances were generated. As a result, a total of 400 instances are obtained to test the comparative performance of the APS model vs. the BPS (first numerical experiment) and a total of 80 instances to evaluate the performance of the APS model under different release and due date combinations (second numerical experiment). All instances generated for the BPS and APS models were solved using AMPL and CPLEX 12.6. All computations were performed on a PC with an I7-3770 Processor with 3.40 GHz and 16 GB of RAM.

Due to computational complexity, not all the instances are solvable in reasonable amounts of time; therefore, we set a fixed time limit for the execution of both models. Specifically, the computational runs are set to initiate a time-out sequence at 3600 seconds (1 hour). Although a time limit is present, we observe that the amount of time to produce a feasible integer solution for the instances that time-out can vary

depending on the model used and the size of the instance. We compare the optimality gap and convergence between the two models for all problem instances.

Table 5.3: Parameter Generating Expressions

Model Parameters	Expressions
Time horizon - T	$r_{last} + \text{round}(n/\alpha_x)$
Release date - r_i	$\text{UNIF}(1, \text{round}(n/2))$ Condensed Release
	$\text{UNIF}(1, 2n)$ Sparse Release
Due date - d_i	$r_i + \lceil 1/\alpha_x \rceil + \text{UNIF}(0, a_2)$ Constant Slack
	$r_i + \lceil 1/\alpha_x \rceil + \text{UNIF}(a_2, a_3)$ Prop. Slack
Tardiness penalty - w_i	$\text{UNIF}(1, a_1)$ 1 job/customer
	$\text{UNIF}(1, \lceil a_1/2 \rceil)$ 2 Jobs/customer
	$\text{UNIF}(1, \lceil a_1/4 \rceil)$ 3-4 Jobs/customer
	$\text{UNIF}(1, \lceil a_1/8 \rceil)$ 5-8 Jobs/customer
Capacity - α_x	$\text{UNIF}(0.1, 0.9)$
Cost of overtime - c_v	$\text{UNIF}(2a_4, 6a_4)$
Overtime capacity - α_v	$\text{UNIF}(0.1, \alpha_x/3)$ if $\alpha_x \geq 0.333$ else $\text{UNIF}(0.1, \alpha_x)$
	$a_1 = n$, $a_2 = 2$, $a_3 = \lceil 2/\alpha_x \rceil$, $a_4 = \overline{w}/\alpha_x$

5.5.1 Performance Testing: The BPS vs. APS model

We investigate the results of all all 400 instances; 100 of them tested under both models, and 300 tested only under the APS model for a total of 500 runs. In our numerical analysis we tracked the number of instances that were solved optimally by each model within the given time frame. The BPS model was able to find the optimal solution in 45% of the instances (45 of the 100 instances) while the APS model produced optimal solutions in 72% of the instances with no aggregation (72 instances out of 100) and 75% over all the instances including aggregation and no aggregation . Figure 5.4 illustrates these observations in more detail. As expected,

in both cases, the number of optimally solved instances decreases as the problem size (*i.e.*, the number of jobs) increases.

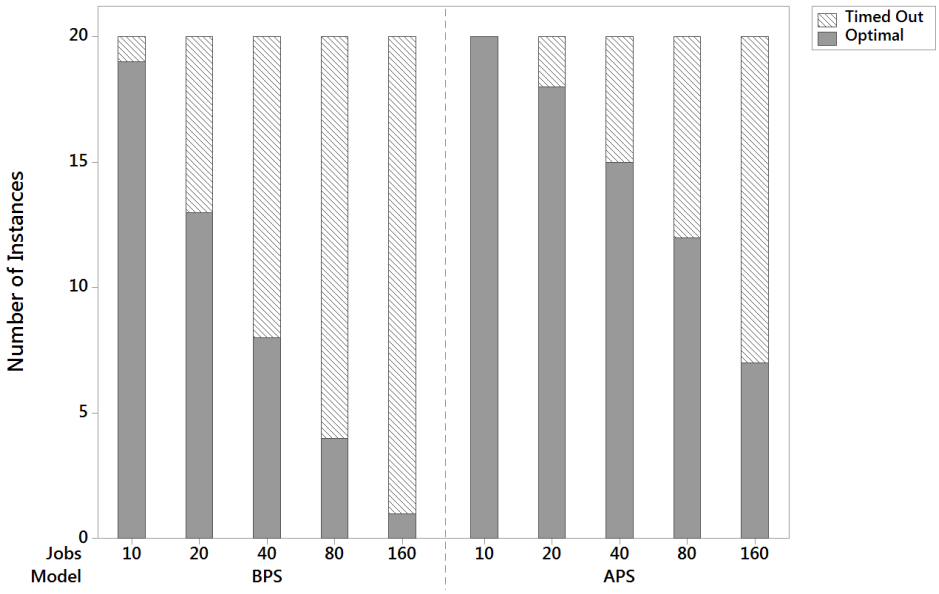

Figure 5.4: Count of optimal and timed-out instances for Group 1 (No aggregation)

While the APS model reported optimal solutions for all instances of the smallest problem size (10 jobs), one of the instances for this problem size did not reach optimality with the BPS model within the preset time limit. The gap between two models in this respect further increases with 20 and more jobs; this is, on average the APS model produced seven more optimal solutions than the BPS model on instances equal or larger than 20 jobs indicating a superior computational performance.

Looking at the impact of aggregation on the performance of the APS model (tables 5.6 through 5.8) we see, as expected, that the number of instances solved optimally gradually increases as there is more aggregation, and the average computational time is reduced. More specifically, the APS model went from 72% of optimally solved instances under no aggregation to 80% of optimally solved instances when aggregation is present at a level of 5 to 8 jobs per customer (Group 4).

The computational time performances and optimality gaps between both models also show significant differences. Measures of computational time performance represented by the minimum, average, and maximum values observed across all problem sizes show that the APS model converges to an optimal or integer feasible solution in a lower average time. Under no aggregation, the average computational time of the APS model is between 49% to 99% faster than the BPS model, and this time is further reduced as aggregation is increased to the APS model. The average computational time improves by an average of 40.1% as aggregation is increased to 5 to 8 jobs per customer. Similarly the optimality gap is represented via the minimum, average and maximum values. The optimality gap results indicate a significant difference in computational time performances where the APS model significantly dominates the BPS model in all instances. We observe that under no aggregation even with relatively small size problems (*i.e.*, 10 jobs), while it takes minutes for the BPS model to reach at optimality, the APS achieves the same result within a fraction of a second on average. As the problem size increases, the convergence performance degrades significantly for the BPS model where the average optimality gap grows up to 19.65% for 160 jobs with a worst case performance of 55.1% gap. On the other hand, the largest observed optimality gap with the APS model is well below 3% and with a worst case performance of 6.23% on one of the instances with no aggregation for 160 jobs. Figure 5.5 summarizes these observations for all job levels. As expected, the performance of the APS model improves in terms of optimality gap as the number of job clusters decreases (*i.e.* as the number of jobs per cluster increases); this is reflected in the last group where 5 to 8 jobs per customer are possible (Table 5.8), and where the APS improved the optimality gaps with respect to the first group by at

least 14.23%. A more detailed description of the specific results for each aggregation group and models is summarized in tables 5.4 through 5.8.

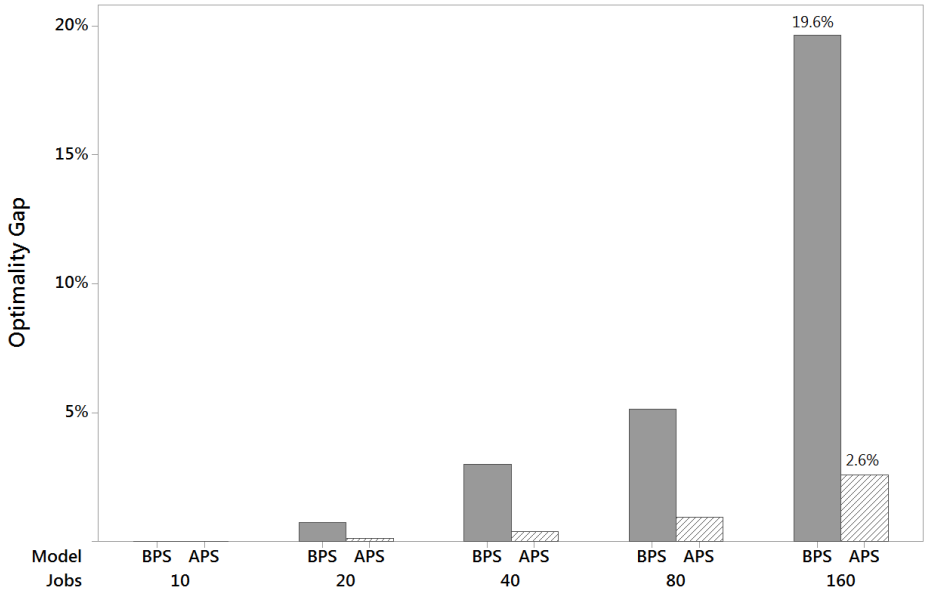

Figure 5.5: Optimality gap by problem size in Group 1 (No aggregation)

Table 5.4: Group 1 - BPS No aggregation (100 instances)

	BPS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	1	0.03	598.4	3606	0.00	0.00	0.01
20	7	0.27	1286	3828	0.00	0.74	4.20
40	12	1.93	2275	3949	0.00	3.01	12.2
80	16	9.91	2999	3651	0.00	5.13	27.5
160	19	852.1	8173	43619	0.00	19.6	55.1

5.5.2 Special Case: No Overtime Option

In this section, we investigate the performance of the BPS and the APS models for the special case where the overtime capacity is zero. This special case of the problem

Table 5.5: Group 1 - APS No aggregation (100 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.030	0.614	3.259	0.00	0.00	0.01
20	2	0.078	373.5	3644	0.00	0.12	1.32
40	5	0.390	1145	4240	0.00	0.40	2.67
80	8	4.774	1623	4341	0.00	0.95	5.09
160	13	16.62	2515	4164	0.00	2.60	6.23

Table 5.6: Group 2 - 2 Jobs per Customer (100 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.031	0.395	1.232	0.00	0.00	0.01
20	2	0.063	363.8	3639	0.00	0.11	1.42
40	4	0.280	784.3	4076	0.00	0.37	2.46
80	8	2.527	1575	4384	0.00	0.86	6.04
160	14	11.46	2694	3961	0.00	3.24	7.79

is denoted by $1|pmtn, r_i, p_i = p| \sum_i w_i l_i$. Although availability of overtime capacity is in general consistent with the real life applications, there may be cases where such option does not exist. Moreover, from the research point of view, we are interested in examining whether the lack of overtime capacity affects the computational performance for the scheduling models under study. To this end we reduce the testing to only 15 instances per scenario, four job levels (10, 20, 30 and 40 jobs), and perform the side by side testing along two aggregation groups used in section 5.5.1 (Group 1 - No aggregation, and Group 2 - 5 to 8 jobs per cluster). In this case the same instances

Table 5.7: Group 3 - 3 to 4 Jobs per Customer (100 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.031	0.215	0.686	0.00	0.00	0.01
20	2	0.047	380.6	3988	0.00	0.17	2.17
40	4	0.249	753.8	3875	0.00	0.30	1.94
80	7	2.246	1410	4204	0.00	1.02	5.28
160	12	22.18	2401	3883	0.00	2.63	6.24

Table 5.8: Group 4 - 5 to 8 Jobs per Customer (100 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.032	0.410	2.465	0.00	0.00	0.01
20	0	0.031	35.44	483.6	0.00	0.00	0.01
40	4	0.187	731.7	3603	0.00	0.17	1.03
80	6	0.375	1303	3967	0.00	0.43	4.60
160	10	5.160	1890	4065	0.00	2.23	6.64

that are used to test the APS model in aggregation Group 2 are also solved under the BPS model to provide a side-by-side view of the performance of both models.

In our analysis, the overtime capacity limit is set to zero, namely, setting $\alpha_v = 0$. The results of these runs are summarized in Tables 5.9 through 5.12. From the 120 instances tested, the BPS model led to optimal solutions in 24 instances (20%) within the 1-hour time frame while the APS model managed to solve 82 out of the 120 (68%) instances optimally. In terms of the optimality gap, the APS model performed significantly better than the BPS, similar to the general case. While the relative gap did not exceed 2.44% in the worst case for the APS model, the convergence performance

for the BPS model, again, degrades significantly as the problem size increases. Our results do not indicate any considerable difference in computational performance between the special case where overtime capacity is not available and the general case that incorporates any positive overtime capacity. Overall, the APS model provides significant gains in computational performance for the total weighted tardiness single machine preemptive scheduling problem with release dates and identical processing times.

Table 5.9: Group 1 - BPS No aggregation (60 instances)

	BPS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	1	0.094	471.7	3606	0.00	0.00	0.00
20	15	3605	4295	5007	0.00	1.55	6.61
30	15	3602	3970	5043	1.94	3.98	7.36
40	15	3600	3805	4191	3.21	5.33	7.37

Table 5.10: Group 1 - APS No aggregation (60 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.043	2.500	11.82	0.00	0.00	0.00
20	4	11.08	1424	3787	0.00	0.63	2.44
30	14	4.071	3539	4105	0.00	1.52	2.30
40	15	3603	3876	4185	0.68	1.57	2.08

5.5.3 Extended Testing for the APS Model

Numerical experiments focused on the APS model show how the performance of the APS model is affected by the arrangement of release and due dates in the

Table 5.11: Group 4 - BPS Model no Aggregation (60 instances)

	BPS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	5	0.125	1262	3608	0.00	0.03	0.44
20	15	3606	3977	5161	0.00	0.32	1.43
30	15	3605	4006	4797	0.16	1.76	4.21
40	15	3605	3765	4065	1.53	3.08	5.10

Table 5.12: Group 4 - 5 to 8 Jobs per Customer on APS Model (60 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.027	0.115	0.617	0.00	0.00	0.00
20	0	0.057	1.532	6.993	0.00	0.00	0.00
30	3	0.793	743.5	3603	0.00	0.06	0.42
40	2	1.674	711.5	3603	0.00	0.04	0.49

schedule; The two levels of problem sizes (20 and 40 Jobs) consistently show that instance complexity increases also as a function of the relative arrangement of the release dates and due dates. Across the release date arrangements it is observed that the number of instances solved optimally decreases when the jobs are released in close proximity to each other (*i.e.* Condensed Release Dates : $r_i = \text{UNIF}(1, \text{round}(n/2))$); in particular the effect of condensed release dates is noted more dramatically under the group of 40 jobs where the number of optimally solved instances goes from an average of 16 (Table 5.16)to an average of 1 (Table 5.15); as expected, the computational time and the optimality gap also change dramatically as the release dates are arranged closer together. On average the computational time triples and the optimality gap quadruples with the condensed arrangement.

Across the due date arrangements a similar pattern is observed however less dramatic; the number of instances solved optimally also decreases when the due dates on the jobs is further constrained by a narrow constant slack (*i.e.* Constant due date Slack : $r_i + \lceil 1/\alpha_x \rceil + \text{UNIF}(0, a_2)$), and the computational time reflects a marginal change as the due date slack is decoupled from the job duration. A more significant impact nonetheless is observed on the optimality gap where the constant due date slack produces about of twice as large gaps as the ones observed on the proportional slack group. Overall, the impact of constrained release and constant due date slack arrangements proves to be consistent with the expected intuitive outcome, and the complexity increases as the schedule is made tighter. A detailed summary of the specific results for each test group is summarized in tables 5.13 through 5.16 where the abbreviation CS stands for *Constant due date Slack* , and PS stands for *Proportional due date Slack*.

Table 5.13: APS Testing - 20 Jobs No aggregation (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	6	1.056	1194	3861	0.01	0.50	2.28
PS	4	0.550	801.9	3818	0.00	0.20	1.66

Table 5.14: APS Testing - 20 Jobs No aggregation (40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	2	0.078	373.5	3644	0.00	0.12	1.32
PS	2	0.109	432.7	3653	0.00	0.06	0.71

Table 5.15: APS Testing - 40 Jobs No aggregation (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	20	3602	3775	4090	0.06	1.70	3.35
PS	18	56.18	3438	4211	0.01	1.10	2.73

Table 5.16: APS Testing - 40 Jobs No aggregation (40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	5	0.390	1145	4240	0.00	0.40	2.67
PS	3	0.172	685.6	4098	0.00	0.25	1.96

5.6 Case Study: Overhaul Scheduling

Numerical experiments done so far show how the performance of the APS model is substantially better than the BPS model. In this section we validate previous observations by using the APS model on a real industry case; we do this by using the model to compute an optimal schedule on the upcoming contracts of a local landing gear overhaul company, and observing the benefits and limitations of using this method in this particular industry.

The business of landing gear overhaul services operates within a very competitive market where failure to meet deadlines represents significant losses for airlines, and in turn loss of business for the overhaul service provider (OSP); the demand of overhaul services has the particularity that is known with anticipation as the time between overhauls (TBO) is dictated by the aircraft manufacturer and enforced by the regulatory agency (FAA); this produces a deterministic set of dates in which the airlines

deliver and expect to receive a landing gear back from an OSP. In a perfect world, the overhaul start and finish dates always stay within with the airline's scheduled time frame; however, capacity limitations from the OSP sometimes require the airline to modify such dates against their financial interests. Commonly, the date in which a landing gear is delivered back to the airline (Overhaul finish date) is considered a hard constraint and is not violated as it represents significant financial losses for airlines whereas the date in which a landing gear is delivered for service to a OSP (overhaul received date) is a soft constraint that can be violated by the OSP, but not without financial loses proportional to the loss of TBO produced to the airline - this is because any reduction to the TBO translates into a loss in the financial return the airline expects from the asset.

In this case study we optimize the overhaul schedule for a local OSP; the schedule consists of 82 jobs expected to be received between the month of November of 2017 and the end of the first quarter of 2019; jobs are unevenly distributed among 4 airlines where the received and due dates are known for each job. The due dates are considered hard constraints while the received dates are soft constraints; this is, the OSP can request the airline to deliver a landing gear earlier than expected, but not without incurring into earliness penalties. We assume that the processes in the OSP shop can be accurately represented by a preemptive single machine model as the majority of the overhaul processes are sequential and can be interrupted in favor of higher priority work. The objective is to produce an overhaul schedule that minimizes the cost of earliness and overtime incurred by the OSP. Although the problem configuration in this case study is not strictly a tardiness problem, the earliness configuration of the OSP schedule in fact represents a mirror case of a tardiness problem;

using this problem property, we transform the OSP received and due dates into their mirror equivalents to generate the equivalent tardiness problem and solve it via the APS model. The mirror transformation is done about the value for the magnitude of the time horizon T, and mirror values for the received, due, and finish dates (r_m, d_m f_m respectively) are obtained from the due, received, and start dates (d_a, r_a, s_a respectively) from the actual OSP schedule. Figure 5.6 illustrates two mirror schedules transformed by Expressions 5.70 through 5.72.

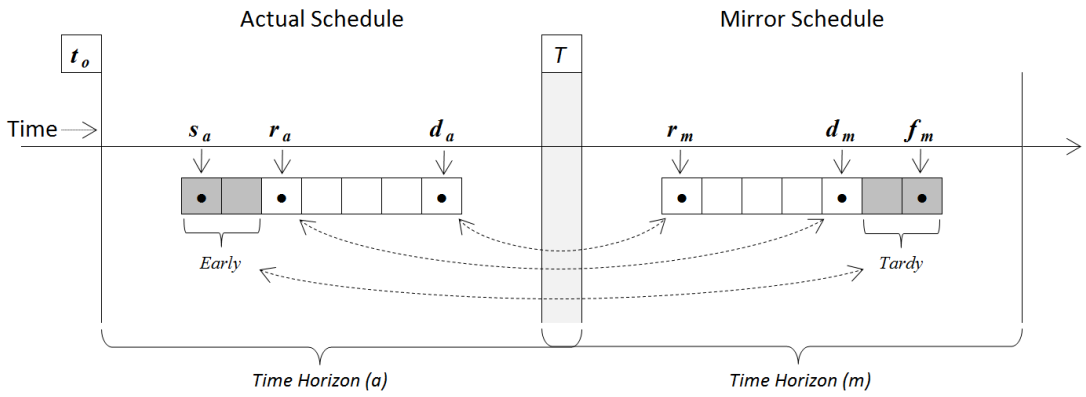

Figure 5.6: Schematic of the date correspondence between actual and mirror schedules

$$r_m = T - d_a + 1 \tag{5.70}$$

$$d_m = T - r_a + 1 \tag{5.71}$$

$$f_m = T - s_a + 1 \tag{5.72}$$

The processing time for all jobs is assumed to be constant; as the OSP uses the same amount of time on performing standard overhaul services on equivalent landing gears - in this case all the 82 jobs are known to belong to equivalent aircrafts. In order

to quantify the earliness penalties incurred by the OSP, we compute the prorated cost of the overhaul service over the number of days lost from the TBO due to an early start of the overhaul, and we give additional weight to such penalty proportional to the size of the business given by the airline *i.e.* the number of jobs present for the same airline. Using the same nomenclature as in the APS model description we let the index i represent each job, J represent the set of job clusters (or airlines) and M_j is the subset of jobs that are aggregated under cluster j ($j \in J$) where $|M_j|$ represents the number of jobs from airline j. The parameter generating expression for the earliness penalty as well as other parameters used in this case are listed in Table 5.17 the superscripts $(^m)$ and $(^a)$ are used to denote parameters corresponding to the mirror and actual schedule respectively.

Table 5.17: Parameters Generating Expressions - Case Study

Mirror Model Parameters	Expressions
Time horizon - T^m	755 Periods
Release date - r^m_{ji}	$T - d^a_{ji} + 1$; $\forall j, i : i \in j$
Due date - d^m_{ji}	$T - r^a_{ji} + 1$; $\forall j, i : i \in j$
Tardiness penalty (weight) - w_j	$(OC_j/TBO_j) \times \lvert M_j \rvert \ \forall j \in J$
	\$2,288 - For airline 1 $\lvert M_1 \rvert = 41$
	\$568 - For airline 2 $\lvert M_2 \rvert = 16$
	\$1,216 - For airline 3 $\lvert M_3 \rvert = 13$
	\$828 - For airline 4 $\lvert M_4 \rvert = 12$
Allowed Regular Time Capacity - α_x	0.1905 - Based on 16 hours regular time
Cost of overtime - c_v	31,500 - Per unit of overtime capacity used
Allowed Overtime capacity - α_v	0.0476 - Based on 4 hours of overtime
	t_0 = Present day, OC = Overhaul Cost

Results from the optimization produced an optimal overhaul schedule in less than four seconds of run time. The model data was processed using AMPL and CPLEX

12.6. All computations were performed on a PC with an I7-3770 Processor with 3.40 GHz and 16 GB of RAM. The optimal schedule indicates that earliness penalties and overtime capacity are required in order to fulfill the orders expected during the end of 2017 and the first quarter of 2019; in particular, a total cost in earliness penalties of $12,008.00 is required to cover the six periods lost from the TBO of two of the airlines, and a total of $17,892.00 of overtime capacity is necessary to avoid further earliness penalties. A breakdown of the optimization costs is presented in Table 5.18.

Table 5.18: Cost Breakdown - Optimal Overhaul Schedule

Cost	Breakdown
Total Schedule Cost	$\$29,900.00$
Cost of Earliness Penalties	$\$12,008.00$
	Five early periods for Airline 1 $\$11,440$
	One early period for Airline 2 $568
Cost of Overtime Capacity Used	$\$17,892.00$
	$\$7,493.85$ on Airline 1
	$\$4,400.55$ on Airline 2
	$0.00 on Airline 3
	$\$5,997.60$ on Airline 4

In addition to provide an estimate of the upcoming costs due to penalties and overtime, the optimization exercise provides the OSP a picture of the expected capacity utilization. Figures 5.7 through 5.9 illustrate the regular and overtime capacity utilization as well as the periods in which earliness penalties are expected. The arrow pointing down in Figures 5.8 and 5.9 corresponds to the time period in which regular time work is assigned to the first overhaul job.

In general, the APS modeling approach proves to be a quick and versatile tool geared to assist the OSP management team with specific information that can be used

to make decisions and prepare for upcoming costs; however, the use of aggregation has a small inherent drawback when it comes to provide detailed information on the capacity allocated to every individual job; this is, the use of aggregation produces a job schedule determined by the capacity allocated to each customer and not to each job; therefore, for more tactical decisions and specific information on the start and finish dates of each job, additional post-processing (dissaggregation) of the results is required.

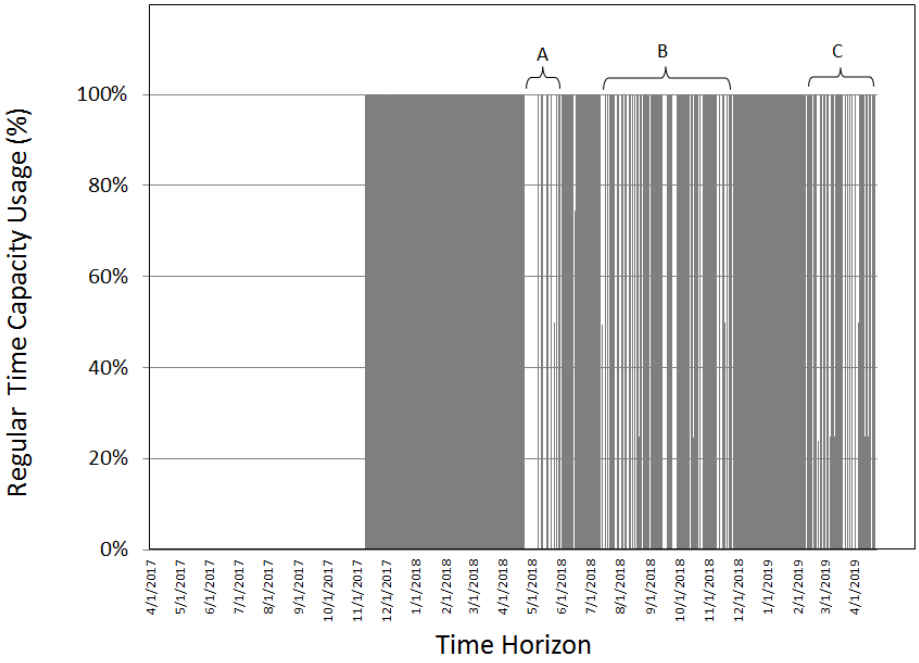

Figure 5.7: Optimal regular time capacity utilization - A, B, C indicate periods with low demand

5.7 Conclusions

We propose two mixed integer programming models for the single machine preemptive scheduling problem that minimizes a composite cost function of total tardiness and overtime capacity utilization. Following the convention of scheduling, we

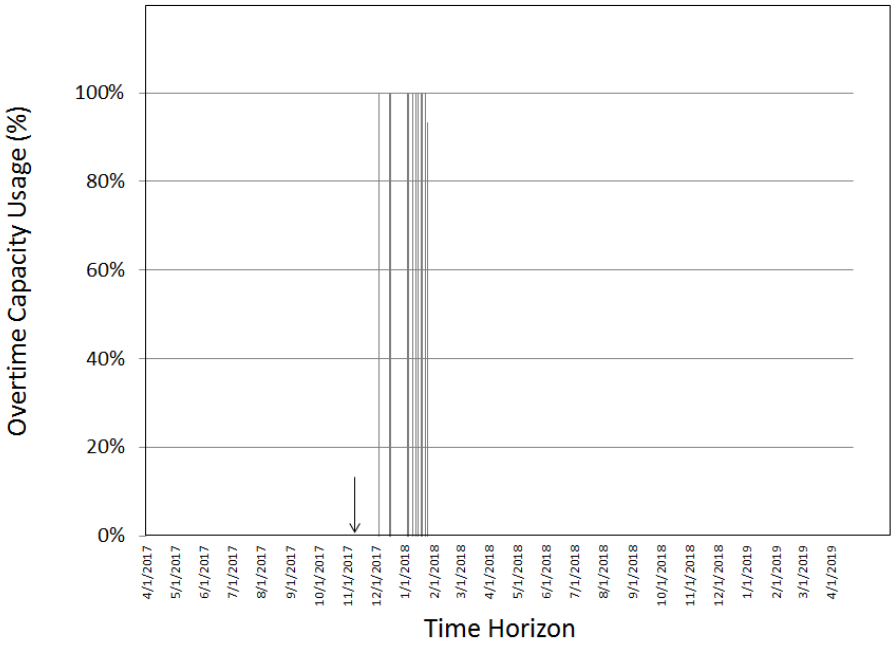

Figure 5.8: Optimal overtime capacity utilization

first build a model using binary variables that identifies completion times of a finite set of jobs with varying release times, due dates, and identical processing times. In this approach, which we referred to as the Binary Preemptive Scheduling (BPS) model, jobs are allocated to available time intervals offered by the existing regular and overtime capacity. The second model, which we call the Aggregate Preemptive Scheduling (APS) model, employs the capacity allocation view and adopts the aggregate planning approach. In this case, existing capacity is distributed among jobs, where jobs are aggregated across their weights as workloads to be fulfilled. With this approach, all binary variables can be eliminated.

In order to test and compare both models in computational efficiency, we evaluated both models using analytical methods and designed and performed a numerical analysis. Our analysis aimed at determining the computational performance of both models with respect to the number of jobs and the job aggregation (clustering) levels.

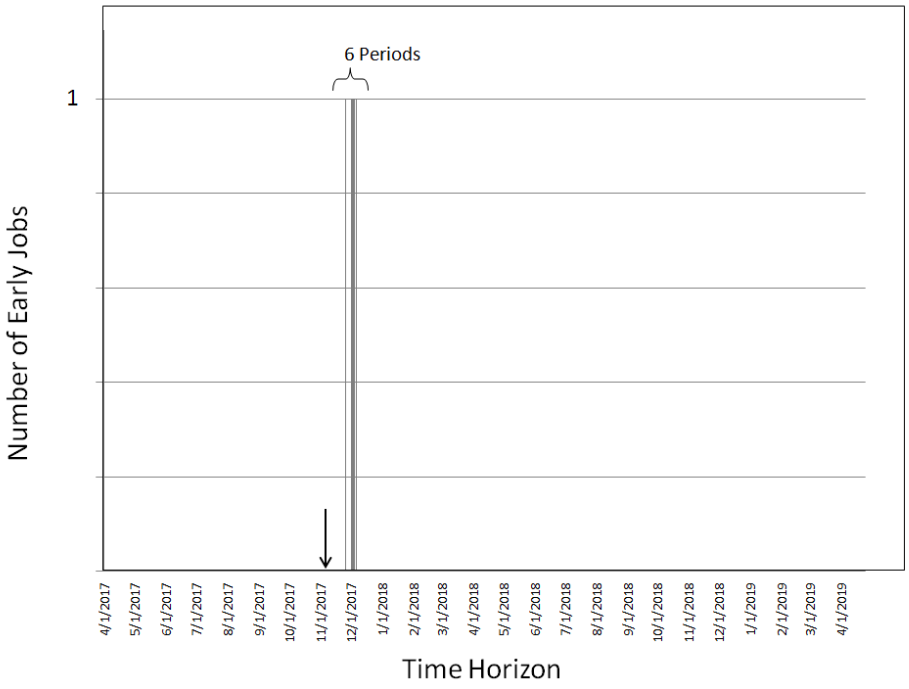

Figure 5.9: Optimal earliness by period

We observed that the BPS model was outperformed with clear lead in all instances. Results of our analysis demonstrate improved lower bounds, strong computational performance and convergence of the APS model and its patent dominance to the BPS model. Overall, the study reveals that APS model is a promising tool for generating optimal or near-optimal schedules and capacity allocation plans reasonably quick for real industry applications.

CHAPTER 6

Advanced Models for Tardiness Problems

6.1 Overview

In this chapter, we explore a set of diverse tardiness related measures in their most general form (*i.e.* with variable processing times). For each one of the measures we present, we compare the performance of two preemptive modeling approaches: a conventional modeling approach side by side to its corresponding advanced model. From the conventional modeling standpoint we use binary variables that keep track of the finish times for all jobs, while adjusting the model to account for the different tardiness related measures; similarly, from the advanced modeling standpoint we apply the aggregate planning paradigm to the different tardiness related problems while maintaining equivalency with its corresponding conventional model. The tardiness related measures we explore include: Total Weighed Tardiness (TWT), Total Weighted Completion (TWC), Total Weighted Earliness and Tardiness (TWET) and Total Weighted Number of Tardy Jobs (TWNTJ). We demonstrate with numerical experimentation and analytic methods that the aggregate modeling approach provides a more efficient platform for model performance therefore representing a promising

tool for generating optimal or near-optimal schedules reasonably quickly for industry applications.

6.2 The Total Weighted Tardiness Problem (TWT)

6.2.1 The BPS Model for TWT

The general BPS optimization model aims to find an optimal schedule that minimizes the total cost of weighted tardiness for a given set of jobs by determining work allocations, finish dates, and tardiness. The nomenclature for the decision variables of the model is given in Table 6.1.

Table 6.1: Decision Variables of the BPS Model Applied to the TWT Problem

x_{it}	Fraction of a job i assigned during regular time at period t
y_{it}	Binary variable: 1 if job i finishes on period t, 0 otherwise
f_i	Finish date for job i
l_i	Number of late periods on job i

The BPS model involves binary, continuous, and discrete decision variables. The binary variables y_{it} track the completion time of jobs. While the continuous variables determine the amount of allocations of regular capacity x_{it} that are necessary to complete the jobs; the discrete variables are needed to evaluate the finish date f_i as well as the number of late periods l_i. The tardiness cost for job i is captured by $w_i l_i = w_i \times \max\{0, f_i - d_i\}$. Consequently, we can write down the mathematical model as follows:

$$\text{minimize: } \sum_{i=1}^{n} w_i l_i \tag{6.1}$$

subject to:

$$\sum_{t=r_i}^{T} y_{it} = 1, \qquad \forall i \in I \tag{6.2}$$

$$f_i = \sum_{t=1}^{T} t y_{it}, \qquad \forall i \in I \tag{6.3}$$

$$l_i \geq f_i - d_i, \qquad \forall i \in I \tag{6.4}$$

$$\sum_{i=1}^{n} x_{it} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.5}$$

$$\sum_{t=r_i}^{T} x_{it} = p_i, \qquad \forall i \in I \tag{6.6}$$

$$\sum_{t=r_i}^{\gamma} x_{it} \geq p_i y_{i\gamma}, \qquad \forall i \in I,\ \forall \gamma \in \{r_i, \ldots, T\} \tag{6.7}$$

$$x_{it} \in [0, 1]\,; \quad f_i,\ l_i \in \mathbb{Z}^+; \quad y_{it} \in \{0, 1\} \tag{6.8}$$

The objective function given in (6.1) minimizes the total cost of weighted tardiness. As mentioned above, the weight for job i, w_i, maps the unit cost of tardiness for that job in this representations. The first set of constraints (6.2) ensure that each job is completed by the end of the planning horizon. The second set of constraints (6.3) capture the completion time of each one of the jobs. Given completion times, constraint in (6.4) sets the tardiness for the jobs. Constraints (6.5) enforces the regular capacity limit designated as α_x. Constraint (6.6) allocates all the work required p_i to complete a job across time periods following its release date, and (6.7) ensures that jobs are not deemed as complete before all required work is done. Solution to the

above model provides an optimal preemptive schedule for a given set of jobs with different release and due dates.

The above model is the product of a conventional modeling approach for the scheduling problems. As expected, the complexity introduced by the combination of the binary outcomes adversely affects the computational performance. To improve the computational efficiency, we employ an approach borrowed from the aggregate planning and develop an alternative model that eliminates the binary variables.

6.2.2 The APS Model for TWT

The proposed aggregate preemptive scheduling (APS) model eliminates the binary variables by aggregating all jobs that have the same weights into a single mutually exclusive set. From practical perspective, each set represents the work orders of a single customer, where different customers carry different significance for the processor. We let J represent the set of job clusters (or customers) and M_j is the subset of jobs that are aggregated under cluster j ($j \in J$). We assume that the jobs in set M_j are sequenced in non-decreasing order of their due dates.

As mentioned above, each cluster in set J consists of jobs that have the same weight. For example, for $n = 6$ jobs, if a weight vector is given by $\mathbf{W} = \{2, 1, 5, 2, 1, 1\}$ for jobs 1 through 6 respectively then the first job and fourth job belong to the same cluster. Likewise, second, fifth, and sixth jobs are aggregated into a single cluster, and third job falls in a separate cluster. Thus, the set of clusters becomes $J = \{1, 2, 3\}$ with the new aggregated weight vector $\overline{\mathbf{W}} = \{1, 2, 5\}$ that maps the weights of job clusters 1, 2, and 3 respectively. Also suppose that the due-date vector

is $\mathbf{d} = \{6, 7, 6, 8, 7, 9\}$ for jobs 1 to 6 respectively in this above example. Consequently, $M_1 = \{2, 5, 6\}$, $M_2 = \{1, 4\}$, and $M_3 = \{3\}$.

While the BPS model employs a one-dimensional array to specify the due date for each job, the APS model requires a three-dimensional array for each job cluster in order to indicate the dates on which its jobs are due. We introduce D_{jit} which denotes the due period t of the job i belonging to cluster j. In the above example $D_{1,1,7} = 1$ since the first job in cluster 1 is due at time 7. Thus, $D_{1,2,7} = D_{1,3,9} = 1$, and $D_{1it} = 0$ for all other (i, t) pairs.

We modify the decision variables in accord with the proposed approach (Table 6.2). Similar to BPS model, we define X_{jit} to track the work assigned to each job; however, in this model X_{jit} does not represent the fraction of work required to complete a single job, but instead it represents the percent of work required to complete the full amount of work corresponding to a job P_{ji}. We introduce two new variables, namely E_{jit} and B_{jit}, that capture the percent of the completed work before the due date and the incomplete work past due for job i in cluster j at time t, respectively. These two variables are mapped to the late periods L_{jit}.

Table 6.2: Decision Variables of the APS Model Applied to the TWT Problem

X_{jit}	Percent of work assigned to job i in cluster j at period t
E_{jit}	Percent of work completed by the deadline for job i in cluster j in period t
B_{jit}	Percent of unfulfilled work after the deadline for job i in cluster j in period t
L_{jit}	Late periods corresponding to job i in cluster j

$$\text{minimize} \sum_{j \in J} \sum_{i \in M_j} \sum_{t=1}^{T} \overline{w}_j L_{jit} \tag{6.9}$$

subject to:

$$B_{ji1} - E_{ji1} = D_{ji1} - X_{ji1}, \qquad \forall j \in J, \forall i \in M_j \tag{6.10}$$

$$B_{jit} - E_{jit} = D_{jit} + B_{jit-1} - X_{jit} - E_{jit-1}, \quad \forall j, i; \forall t \in \{2, 3, ., T\} \tag{6.11}$$

$$\sum_{j \in J} \sum_{i \in M_j} P_{ji} X_{jit} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.12}$$

$$\sum_{t \in T} X_{jit} \leq 1, \qquad \forall j \in J; \forall\ i \in M_j \tag{6.13}$$

$$\sum_{t=1}^{r_{ji}-1} X_{jit} \leq 0, \qquad \forall j \in J, \forall i \in M_j \tag{6.14}$$

$$L_{jit} \geq B_{jit}, \qquad \forall j \in J, \forall i \in M_j \forall\ t \in \{1, 2, \ldots, T\} \tag{6.15}$$

$$X_{jit},\ B_{jit},\ E_{jit} \in [0, 1], \qquad \forall j \in J, \forall t \in \{1, 2, \ldots, T\} \tag{6.16}$$

$$L_{jit} \in \mathbb{Z}^+, \qquad \forall j \in J \forall t \in \{1, 2, \ldots, T\} \tag{6.17}$$

In the above model, the first two constraints are the work balance equations which capture the fraction of late work for each job in each cluster. Constraint (6.12) enforce the capacity limit. Constraint (6.13) ensures that no job is assigned more work than what is required to complete 100% of the work. Constraint (6.14) ensures that work is not assigned to a job on periods prior to its release date. Constraint (6.15) captures each one of the periods in which a job is late. Similar to the BPS model, solution to the above model provides a feasible optimal preemptive schedule for a given set of jobs with different release and due dates.

6.2.3 Computational Experiments

We design and perform numerical analysis so as to evaluate the computational performance of the BPS and the APS models for total weighted tardiness. We aim to determine the computational performance of both models with respect to the size of the instance (*i.e.* number of jobs). Our goal is to demonstrate that the proposed APS model significantly improves the computational performance. To this end we perform two sets of numerical experiments that test both models under different conditions using the number of optimal solutions, computational time and optimality gap as reference measures. In our first numerical experiment we consider five problem sizes with 10, 20, 40, 80, and 160 jobs which we solve side by side using each model. Following the first numerical tests, we proceed with numerical experiments focused only on the APS model. in particular, we use numerical tests to explore how the performance of the APS model changes depending on how the release and due dates are arranged in the schedule; with this aim we produced four scenarios in which we test the combinations between two levels of release dates (i.e. Condensed Release Dates and Sparse Release Dates) and two levels of due dates determined by the slack added to the jobs (i.e. Constant Slack and Proportional Slack). The four resulting scenarios are tested on the APS model for 20 and 40 jobs. The decision to evaluate the APS model only under the arrangement of release and due dates, on one side, follows observations made by [33] and [24] on the effect of the proximity and arrangement of the release and due dates on the complexity of the instance, and on the other, is mostly the result of multiple observations made over extensive testing of the effect of different schedule parameters on the performance of the models; The effect of other parameters like α_x, and T are implicitly tested under the different job levels, and other

parameters like α_v and w_i do not demand additional testing as their relationship to the complexity of the instances is more intuitively understood.

To produce the two phases of numerical experiments described above we use a set of expressions to generate all the parameters necessary to create the problem instances. To be consistent with the existing literature, we borrowed the expressions from earlier research [20] and modified them to our context. Main random variables in the system are generated by a uniform distribution in order to enable diversity across all instances. To avoid trivial cases, the number of jobs n and the maximum allowed regular time α_x are preset and incorporated as input parameters in the problem instance generation process. Table 6.3 lists all the specific parameter generating expressions used in our analysis. For each scenario combination, 20 instances are generated. As a result, a total of 400 instances are obtained to test the comparative performance of the APS model vs. the BPS (first numerical experiment) and a total of 80 instances to evaluate the performance of the APS model under different release and due date combinations (second numerical experiment). All instances generated for the BPS and APS models are solved using AMPL and CPLEX 12.6. All computations are performed on a PC with an I7-3770 Processor with 3.40 GHz and 16 GB of RAM.

Due to computational complexity, not all the instances are solvable in reasonable amounts of time. Therefore, we set a fixed time limit for the execution of both models. Specifically, the computational runs are set to initiate a time-out sequence at 3600 seconds (1 hour). Although a time limit is present, we observe that the amount of time to produce a feasible integer solution for the instances that time-out can vary depending on the model used and the size of the instance.

Table 6.3: Parameter Generating Expressions

Model Parameters	Expressions
Time horizon - T	$r_{last} + \text{round}(n/\alpha_x)$
Release date - r_i	$\text{UNIF}(1, \text{round}(n/2))$ Condensed Release
	$\text{UNIF}(1, 2n)$ Sparse Release
Due date - d_i	$r_i + \lceil 1/\alpha_x \rceil + \text{UNIF}(0, a_2)$ Prop. Slack
	$r_i + \lceil 1/\alpha_x \rceil + \text{UNIF}(0, a_3)$ Constant Slack
Tardiness penalty (weight) - w_i	$\text{UNIF}(1, a_1)$
Capacity - α_x	$\text{UNIF}(0.15, 0.5)$
Required Work - p_i	$\text{UNIF}(0.6, 1.4)$
	$a_1 = n$, $a_2 = \lceil \overline{p}_i/\alpha_x \rceil$, $a_3 = 2$

We compare the optimality gap and convergence between the two models for all problem instances. For all the numerical experiments the lower bound from the APS model is used as reference to produce the optimality gaps in both models; this is necessary due to the tendency of the BPS model to generate artificially small lower boundaries even when the reached feasible solution is actually close to the optimal. The same test methodology described above is applied to all tardiness related tests, while using the same problem instances across all problems; this is, the same set of instances used to test the models under the TWT problem are again used for the TWC, TWET and TWNTJ. In the particular case of the TWC problem the due date of the instances is modified and made equal to the release dates in order to keep the same model structure for weighted tardiness while solving a weighted completion problem. With the exception of the extended testing on the APS models all instances are produced under the criterion established for Sparse Release Dates so as to produce more realistic combination of job release dates over the schedule.

6.2.4 Computational Results

We investigate the results of all 100 instances on each model (200 runs total). In our numerical analysis we tracked the number of instances that were solved optimally by each model within the given time frame. The BPS model was able to find the optimal solution in 18% of the instances (18 of the 100 instances) while the APS model produced optimal solutions in 40% (40 instances out of 100). Figure 6.1 illustrates these observations in more detail separated by the release date scenarios. As expected, in both cases, the number of optimally solved instances decreases as the problem size (*i.e.*, the number of jobs) increases.

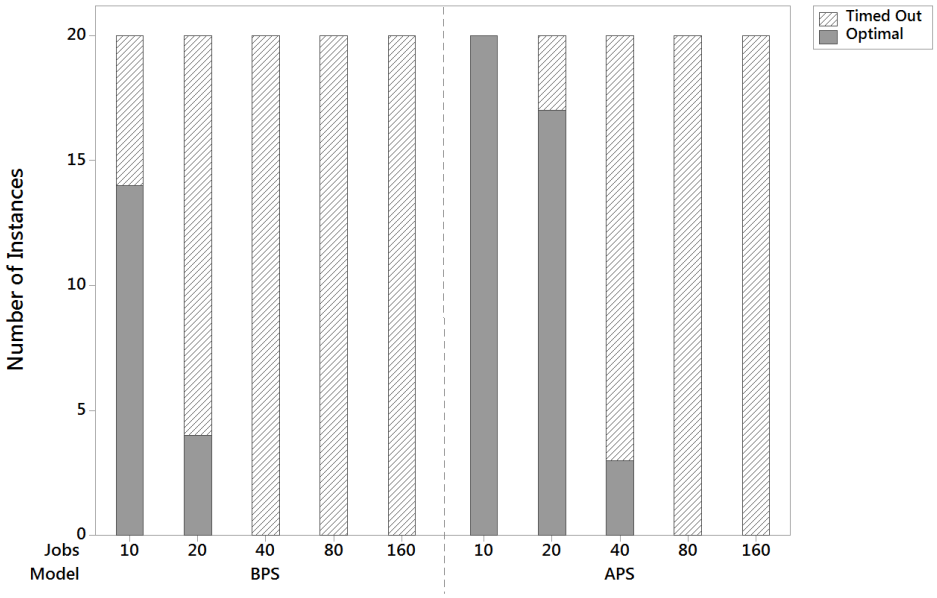

Figure 6.1: Count of optimal and timed-out instances under the TWT problem

While the APS model reported optimal solutions for all instances of the smallest problem size (10 jobs), some portion of the instances for this problem size did not reach optimality with the BPS model within the preset time limit. The gap between two models in this respect further increases with 20 jobs. The BPS model provided

no optimal solutions for the instances with 40 and 80 jobs. Whereas, the APS model was able to reach optimality on some instances with 40 jobs within the 1 hour time frame indicating a superior computational performance.

Next, we compare the computational time performances and optimality gaps between both models. Measures of computational time performance include the minimum, average, and maximum number of seconds used to reach an optimal solution; similarly the optimality gap is represented via the minimum, average and maximum percent error as calculated from the lower boundary produced by the APS problem relaxation. The comparison is performed for each instance size and summarized in tables 6.4 and 6.5.

The results indicate a significant difference in computational time performances where the APS model dominates the BPS model in all instances. We observe that even with relatively small size problems (*i.e.*, 10 jobs), while it takes several minutes for the BPS model to reach at optimality, the APS achieves the same result within a matter of seconds on average. As the problem size increases, the convergence performance degrades significantly for the BPS model. The average optimality gap stays above 9.0% for more than 20 jobs with a worst case performance of 91.2% gap in cases where a feasible integer solution was not found by the solver after a period of one hour. By contrast, the average optimality gap with the APS stays under 28% even for instances of 160 jobs, and for instances of 80 jobs or less the maximum optimality gap resulting from any instance is well below 10%.

Table 6.4: Results for the BPS Model on the TWT Problem (100 instances)

	BPS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	6	0.20	1207	3611	0.00	0.06	0.95
20	16	5.15	3020	4027	0.00	1.99	10.1
40	20	3602	3638	4118	0.98	9.19	18.0
80	20	3600	3664	4076	13.9	27.1	48.2
160	20	3609	3612	3616	85.0	88.7	91.2

Table 6.5: Results for the APS Model on the TWT Problem (100 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.031	9.95	135.2	0.00	0.00	0.00
20	3	0.717	685.5	3820	0.00	0.42	3.67
40	17	356.4	3573	4420	0.00	2.99	5.53
80	20	3601	4085	5110	2.17	5.28	8.68
160	20	3601	3687	4189	1.71	27.68	95.15

6.2.4.1 Extended Testing for the APS Model

Numerical experiments focused on the APS model implemented on the TWT problem show how its performance is affected by the arrangement of release and due dates in the schedule; The two levels of problem sizes (20 and 40 Jobs) consistently show that instance complexity increases also as a function of the relative arrangement of the release dates; specifically, in the case of 20 jobs, the total number of instances solved optimally drops from 30 to 12, and in the group of 40 jobs (Tables 6.8 and 6.9) this metric goes from a total of 3 to 0; this is, in both groups the number of instances solved optimally decreases when the jobs are released in close proximity to

each other (*i.e.* Condensed Release Dates : $r_i = \text{UNIF}(1, \text{round}(n/2))$). As expected, the average computational time also changes as the release dates are arranged closer together; this is more noticeable in the 20-jobs group than in the 40-jobs group due to the larger number of instances solved optimally. In terms of the optimality gap, the results observed for the group with 20 jobs fall in line with the performance described above where optimality gaps are smaller when its release dates are spread apart; however, the optimality gaps on the 40-jobs group do not fit the same pattern, and the optimality gap appears to increase as the release dates are separated.

From the perspective of the due date arrangements the total number of instances solved optimally slightly decreases when the due dates on the jobs are constrained by a narrow constant due date slack, although this is only noticeable under the 20-job scenario; similarly the computational time and optimality gaps reflect only a marginal change as the due date slack is decoupled from the job duration, and a noticeable change is only noticed in the 20-jobs group where the total average optimality gap (including condensed and sparse release values) goes from 0.8% under proportional slack to 1.03% under constant slack. A more detailed summary of the specific results for each test group is summarized in tables 6.6 through 6.9.

Table 6.6: APS Extended Testing - Condensed Release Dates on 20 Jobs (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	13	89.28	2805	3886	0.00	1.20	3.29
PS	15	329.2	3188	4077	0.00	1.17	3.59

Table 6.7: APS Extended Testing - Sparse Release Dates on 20 Jobs (40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	7	0.751	1358	4031	0.00	0.87	3.82
PS	3	0.717	685.5	3820	0.00	0.42	3.67

Table 6.8: APS Extended Testing - Condensed Release Dates on 40 Jobs (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	20	3614	3941	4180	1.17	1.78	2.85
PS	20	3601	3920	4113	1.17	1.83	2.35

6.2.5 Special Case: Overtime Available

As an extension of the TWT, here we compare the conventional and advanced modeling approaches in cases when the use of overtime capacity is available; we refer to this tardiness measure as TWTOT. The models used in this section correspond to the BPS and APS models used in sections 6.2.1 and 6.2.2, which have been updated to account for the cost and use of overtime capacity; this is, the models contain a second term in the objective function corresponding to the cost of overtime use and the work constrains are modified to account for the allocated overtime capacity. The implementation of each model to the TWTOT problem is as follows.

Table 6.9: APS Extended Testing - Sparse Release Dates on 40 Jobs (40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	17	168.6	3572	4466	0.00	3.20	10.34
PS	17	356.4	3573	4420	0.00	2.99	5.53

BPS Model:

$$\text{minimize: } \sum_{i=1}^{n} w_i l_i + c_v \sum_{iinI} \sum_{t=1}^{T} v_{it} \tag{6.18}$$

subject to:

$$\sum_{t=r_i}^{T} y_{it} = 1, \qquad \forall i \in I \tag{6.19}$$

$$f_i = \sum_{t=1}^{T} t y_{it}, \qquad \forall i \in I \tag{6.20}$$

$$l_i \geq f_i - d_i, \qquad \forall i \in I \tag{6.21}$$

$$\sum_{i=1}^{n} x_{it} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.22}$$

$$\sum_{i=1}^{n} v_{it} \leq \alpha_v, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.23}$$

$$\sum_{t=r_i}^{T} (x_{it} + v_{it}) = p_i, \qquad \forall i \in I \tag{6.24}$$

$$\sum_{t=r_i}^{\gamma} (x_{it} + v_{it}) \geq p_i y_{i\gamma}, \qquad \forall i \in I,\ \forall \gamma \in \{r_i, \ldots, T\} \tag{6.25}$$

$$x_{it}, v_{it} \in [0, 1]\,; \quad f_i,\ l_i \in \mathbb{Z}^+; \quad y_{it} \in \{0, 1\} \tag{6.26}$$

APS Model:

$$\text{minimize} \sum_{j\in J}\sum_{i\in M_j}\sum_{t=1}^{T}\overline{w}_j L_{jit} + c_v \sum_{j\in J}\sum_{i\in M_j}\sum_{t=1}^{T} P_{ji}V_{jit} \tag{6.27}$$

subject to:

$$B_{ji1} - E_{ji1} = D_{ji1} - K_{ji1}, \qquad \forall j \in J, \forall i \in M_j \tag{6.28}$$

$$B_{jit} - E_{jit} = D_{jit} + B_{jit-1} - K_{jit} - E_{jit-1}, \quad \forall j, i; \forall t \in \{2, 3, ., T\} \tag{6.29}$$

$$K_{jit} = X_{jit} + V_{jit}, \qquad \forall j \in J, \forall i \in M_j \forall\ t \in \{1, 2, \ldots, T\} \tag{6.30}$$

$$\sum_{j\in J}\sum_{i\in M_j} P_{ji}X_{jit} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.31}$$

$$\sum_{j\in J}\sum_{i\in M_j} P_{ji}V_{jit} \leq \alpha_v, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.32}$$

$$\sum_{t\in T} X_{jit} + V_{jit} \leq 1, \qquad \forall j \in J; \forall\ i \in M_j \tag{6.33}$$

$$\sum_{t=1}^{r_{ji}-1} X_{jit} + V_{jit} \leq 0, \qquad \forall j \in J, \forall i \in M_j \tag{6.34}$$

$$L_{jit} \geq B_{jit}, \qquad \forall j \in J, \forall i \in M_j \forall\ t \in \{1, 2, \ldots, T\} \tag{6.35}$$

$$X_{jit},\ V_{jit},\ K_{jit},\ B_{jit},\ E_{jit} \in [0, 1], \quad \forall j \in J, \forall t \in \{1, 2, \ldots, T\} \tag{6.36}$$

$$L_{jit} \in \mathbb{Z}^+, \qquad \forall j \in J, \forall t \in \{1, 2, \ldots, T\} \tag{6.37}$$

To stay consistent with the nomenclature we use the same overtime nomenclature used in Chapter 5. Here the variables v_{it} and V_{jit} represent the amount of overtime capacity used; in the case of the BPS model v_{it} corresponds directly to the units of capacity while in the APS model V_{jit} represents the fraction from the total capacity P_{ji} that uses overtime. In both models the parameter c_v represents the cost per

unit of overtime, and α_v corresponds to the maximum overtime capacity that can be allocated on a given period.

For simplicity, the testing performed in this section does not include instances with 160 jobs as these are not expected to reach any optimal solution, and instances with 80 jobs suffice to illustrate the performance of both models under relatively large instances. Numerical analysis done to both models (total of 80 instances - 20 instances per job-size) shows that the BPS model was able to find the optimal solution in 17% of the instances (14 of the 80 instances) while the APS model produced optimal solutions in 69% (55 instances out of 80). As expected, in both cases, the number of optimally solved instances decreases as the problem size (*i.e.*, the number of jobs) increases. While the APS model reported optimal solutions for all instances of 10 and 20 jobs, a large portion of these instances for these problem sizes did not reach optimality with the BPS model within the preset time limit. The gap between two models in this respect is maintained with 40 jobs. The BPS model provided no optimal solutions for the instances with 40 and 80 jobs; whereas the APS model was able to reach optimality on 14 of the instances with 40 jobs and one of the instances with 80 jobs indicating a superior computational performance. these results are summarized in tables 6.10 and 6.11.

The results indicate a significant difference in computational time performances where the APS model dominates the BPS model in all instances. We observe that even with relatively small size problems (*i.e.*, 10 jobs), while it takes several minutes for the BPS model to reach at optimality, the APS achieves the same result within fractions of a second on average. As the problem size increases, the convergence performance degrades significantly for the BPS model. The average optimality gap

reaches 13% for 80 jobs with a worst case performance of 58.5% gap in cases where a feasible integer solution was not found by the solver after a period of one hour. By contrast, the average optimality gap with the APS stays under 4% even for the worst of the instances of 80 jobs.

Table 6.10: Results for the BPS Model on the TWTOT Problem (80 instances)

	BPS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	8	0.110	1683	3606	0.00	0.00	0.01
20	18	24.86	3251	3607	0.00	0.55	2.71
40	20	3600	3627	4074	0.01	2.16	6.69
80	20	3603	3611	3673	1.85	13.27	58.52

Table 6.11: Results for the APS Model on the TWTOT Problem (80 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.078	0.876	2.559	0.00	0.00	0.01
20	0	0.343	23.99	223.9	0.00	0.01	0.01
40	6	6.817	1563	4364	0.00	0.42	2.49
80	19	3062	3838	4485	0.00	1.54	3.90

6.3 The Total Weighted Completion Problem (TWC)

6.3.1 The BPS Model for TWC

The implementation of the BPS model to the TWC problem does not require any changes to the model itself - it is the same model used in Section 6.2.1; however, the

due dates for all the jobs are made equal to the release dates; this way all jobs are essentially late and the minimization process is be determined only by the completion times of each job. The nomenclature for the decision variables of the model is given in Table 6.12.

Table 6.12: Decision Variables of the BPS Model Applied to the TWC Problem

x_{it}	Fraction of a job i assigned during regular time at period t
y_{it}	Binary variable: 1 if job i finishes on period t, 0 otherwise
f_i	Finish date for job i
C_i	Number of periods to complete job i

Similarly to the implementation of the BPS model to the TWT problem, this model implementation involves binary, continuous, and discrete decision variables. The binary variables y_{it} track the completion time of jobs. While the continuous variables determine the amount of allocation of capacity x_{it} necessary to complete the jobs, the discrete variables are needed to evaluate the finish date f_i as well as the number of late periods l_i - in this case C_i. The tardiness cost for job i is captured by $w_i l_i = w_i max\{0, f_i - d_i\}$. Since we allowed $d_i = r_i$ the expression for the tardiness cost becomes equivalent to the completion cost for job i as $w_i C_i = w_i(f_i - r_i)\}$ where C_i represents the periods used to complete job i. Consequently, we can write down the mathematical model as follows:

$$\text{minimize} \sum_{i=1}^{n} w_i C_i \tag{6.38}$$

subject to:

$$\sum_{t=r_i}^{T} y_{it} = 1, \qquad \forall i \in I \tag{6.39}$$

$$f_i = \sum_{t=1}^{T} t y_{it}, \qquad \forall i \in I \tag{6.40}$$

$$C_i \geq f_i - d_i, \qquad \forall i \in I \tag{6.41}$$

$$\sum_{i=1}^{n} x_{it} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.42}$$

$$\sum_{t=r_i}^{T} x_{it} = p_i, \qquad \forall i \in I \tag{6.43}$$

$$\sum_{t=r_i}^{\gamma} x_{it} \geq p_i y_{i\gamma}, \qquad \forall i \in I,\ \forall \gamma \in \{r_i, \ldots, T\} \tag{6.44}$$

$$x_{it} \in [0, 1]\,; \quad f_i,\ C_i \in \mathbb{Z}^+; \quad y_{it} \in \{0, 1\} \tag{6.45}$$

The objective function given in (6.38) minimizes the total cost of weighted completion. As mentioned above, the weight for job i, w_i, maps the unit cost of time used for that job in this representations. The first set of constraints (6.39) ensure that each job is completed by the end of the planning horizon. The second set of constraints (6.40) capture the completion time of each one of the jobs. Given completion times, constraint in (6.41) sets the completion times (also tardiness) for the jobs. Constraints (6.42) enforces the regular capacity limit designated as α_x. Constraint (6.43) allocates all the work required (p_i) to complete a job across time periods following its release date, and (6.44) ensures that jobs are not deemed as completed before all

required work is done. Solution to the above model provides an optimal preemptive schedule for the TWC problem.

6.3.2 The APS Model for TWC

Similarly to the implementation of the BPS model on the TWC problem, the implementation of the APS model to the TWC problem does not require any model changes. All the variables model remain the same with the tardiness variable now being the variable accounting for the completion times of all the jobs; this results from the re-purposing of the due dates after making them equal to the release dates.

The decision variables for this model are listed in Table 6.13. We measure X_{jit} as the percent of the total work P_{ji} required to complete a single job. In this particular model the variables representing the work done before the due date become irrelevant; this is, E_{jit} while B_{jit} captures the percent of the work completed after the release date for job i in cluster j at time t; this variable is also mapped to the periods used in the completion of a job C_{jit}.

Table 6.13: Decision Variables of the APS Model Applied to the TWC Problem

X_{jit}	Percent of work assigned to job i in cluster j at period t
E_{jit}	Percent of work completed by the deadline for job i in cluster j in period t
B_{jit}	Percent of unfulfilled work after the deadline for job i in cluster j in period t
C_{jit}	Work periods corresponding to job i in cluster j

The mathematical model is

$$\text{minimize} \sum_{j\in J}\sum_{i\in M_j}\sum_{t=1}^{T} \overline{w}_j C_{jit} \tag{6.46}$$

subject to:

$$B_{ji1} - E_{ji1} = D_{ji1} - X_{ji1}, \qquad \forall j \in J, \forall i \in M_j \tag{6.47}$$

$$B_{jit} - E_{jit} = D_{jit} + B_{jit-1} - X_{jit} - E_{jit-1}, \quad \forall j, i; \forall t \in \{2, 3, ., T\} \tag{6.48}$$

$$\sum_{j\in J}\sum_{i\in M_j} P_{ji} X_{jit} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.49}$$

$$\sum_{t\in T} X_{jit} \leq 1, \qquad \forall j \in J; \forall\ i \in M_j \tag{6.50}$$

$$\sum_{t=1}^{r_{ji}-1} X_{jit} \leq 0, \qquad \forall j \in J, \forall i \in M_j \tag{6.51}$$

$$C_{jit} \geq B_{jit}, \qquad \forall j \in J, \forall\ t \in \{1, 2, \ldots, T\} \tag{6.52}$$

$$X_{jit},\ B_{jit},\ E_{jit} \in [0, 1], \qquad \forall j \in J \forall t \in \{1, 2, \ldots, T\} \tag{6.53}$$

$$C_{jit} \in \mathbb{Z}^+, \qquad \forall j \in J \forall t \in \{1, 2, \ldots, T\} \tag{6.54}$$

Similarly to the TWT model, the first two constraints are the work balance equations which capture the percent of work done after the release of each job in each cluster. Constraint (6.49) enforce the capacity limit. Constraint (6.50) ensures that no job is assigned more work than what is required to complete 100% of the work. Constraint (6.51) ensures that work is not assigned to a job on periods prior to its release date. Constraint (6.52) captures each one of the periods required to complete a job. Similar to the BPS model, solution to the above model provides a feasible optimal preemptive schedule for the TWC problem.

6.3.3 Computational Experiments

For the testing of the APS and BPS modeling approaches as they are implemented to solve the TWC problem we follow the same methodology used during the testing of the models on the TWT problem - Section 6.2.3. For experimental consistency throughout the research, here we perform the same two sets of numerical experiments and solve the same instances used in Section 6.2; although, part of the extended testing done to the APS model is not really relevant for the TWC problem; this is because the nature of the TWC problem does not depend on the due dates, and the scenarios created to explore the effect of the due date slack do not have an effect on the model performance; nonetheless, we consider pertinent to present this results.

6.3.4 Computational Results

Numerical analysis on all 100 instances (20 for each problem size) on each model (200 runs) revealed the number of instances that were solved optimally by each model within the given time frame. The BPS model was able to find the optimal solution in 20% of the instances (20 of the 100 instances) while the APS model produced optimal solutions in 36% (36 instances out of 100). Figure 6.2 illustrates these observations in more detail. As expected, in both cases, the number of optimally solved instances decreases as the problem size (*i.e.*, the number of jobs) increases.

While the APS model reported optimal solutions for all instances of the smallest problem size (10 jobs), some portion of the instances for this problem size did not reach optimality with the BPS model within the preset time limit. The gap between the two models in this respect is also noticeable with 20 jobs. The BPS model as well

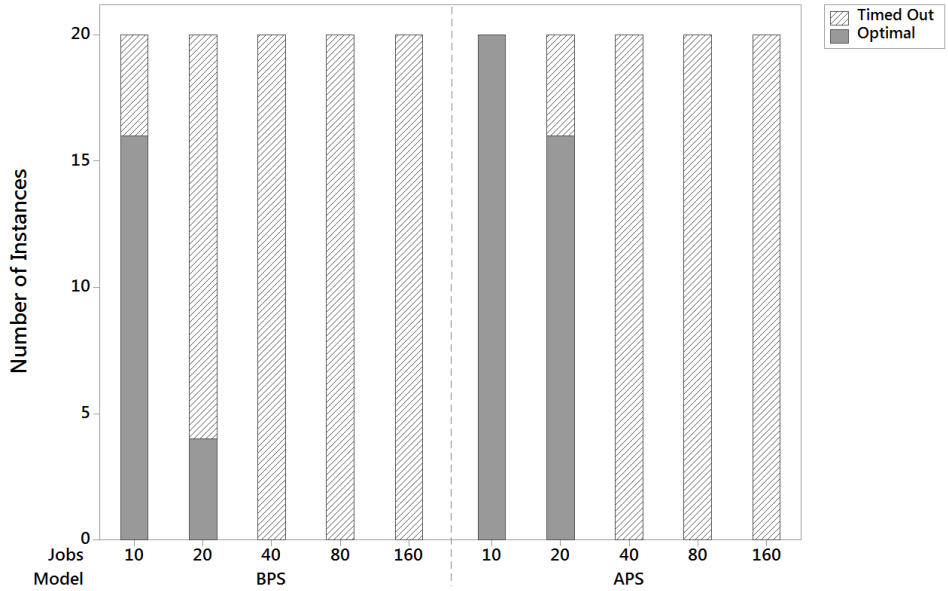

Figure 6.2: Count of optimal and timed-out instances under the TWC problem

as the APS provided no optimal solutions for the instances larger than 20 jobs within the 1 hour time frame. Although the performance difference between the models is not apparent on larger instances (40, 80, and 160 jobs) as compared with respect to the number of time-outs, the percent optimality gap reveals that the superior performance of the APS model is present even under larger size problems. These results are summarized in tables 6.14 and 6.15.

Table 6.14: Results for the BPS Model on the TWC Problem (100 instances)

	BPS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	4	0.14	1033	3640	0.00	0.01	0.23
20	16	11.59	2971	4830	0.00	0.99	8.04
40	20	3602	3628	3989	2.63	6.45	13.03
80	20	3603	3648	3919	5.68	16.8	41.14
160	20*	3620	3636	3652	72.04	75.79	81.25

*Only 4 out of the 20 instances reached a feasible integer solution

Table 6.15: Results for the APS Model on the TWC Problem (100 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.109	24.51	258.7	0.00	0.00	0.00
20	4	3.040	1010	4071	0.00	0.43	3.73
40	20	3623	4031	4417	2.34	3.56	5.57
80	20	3601	3985	4795	2.06	4.65	6.84
160	20	3604	3694	4077	2.94	45.25	96.61

In terms of computational time, we observe that even with relatively small size problems (*i.e.*, 10 jobs), while it takes minutes for the BPS model to reach at optimality, the APS achieves the same result within a fraction of a second on average. As the problem size increases, the convergence performance on the BPS model degrades significantly, and the average optimality gap stays above 16% for more than 40 jobs with a worst case performance of 81.25%. On the other hand, the optimality gap with the APS model remains under 7% on instances of upto 80-jobs with a worst case performance of 96.61% on 160 jobs. In terms of the maximum values reported for the optimality gap with 160 jobs, the BPS model appears to produce slightly lower values than the APS model (81.25% vs. 96.61% respectively); this data artifact is the result of the lack of actual data from the BPS runs for 160 jobs which resulted in only 4 instances with feasible integer solutions and 16 instances for which the time limit was surpassed withouth the solver reaching an integer feasible solution.

6.3.4.1 Extended Testing for the APS Model

Here, the effect of the arrangement of release and due dates in the schedule is explored for the implementation of the APS model on the TWC problem; In terms of

the timed-out instances, the 20-jobs group shows that instance complexity increases also as a function of the relative arrangement of the release dates; in particular, the number of optimally solved instances goes from an average of 29 to an average of 5; as expected, the computational time for this group also changes from an average of 1232 seconds under sparse release dates to an average time-out value of 3449 seconds under condensed release, and the optimality gap increases from an average of 0.53% to 1.14% also from sparse to condensed release respectively. The pattern observed in the 20-jobs group is consistent in showing the higher complexity produced by the condensed release dates; however, this observation cannot be made on the 40-jobs group where virtually all instances timed out without an optimal solution, and the optimality gaps show an increase in value from 1.68% under condensed release to 3.22% under sparse release dates.

From the perspective of the due date arrangements, as expected, no discernible pattern is observed; this is because the data reflects the imperviousness of the TWC problem to the due date slack where no significant change is noticed as the instances were tested under constant or proportional due date slack. A detailed summary of the results for each test group is summarized in tables 6.16 through 6.19.

Table 6.16: APS Extended Testing - Condensed Release Dates on 20 Jobs (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	16	77.35	3249	4142	0.00	1.08	2.64
PS	19	1181	3648	4404	0.00	1.20	2.57

Table 6.17: APS Extended Testing - Sparse Release Dates on 20 Jobs (40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	7	1.493	1455	3927	0.00	0.62	3.10
PS	4	3.040	1010	4071	0.00	0.43	3.73

Table 6.18: APS Extended Testing - Condensed Release Dates on 40 Jobs (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	20	3607	3959	4260	1.13	1.67	2.36
PS	20	3601	3992	4222	1.27	1.68	2.48

6.4 The Total Weighted Earliness and Tardiness Problem (TWET)

6.4.1 The BPS Model for TWET

The BPS optimization model for the total weighted earliness and tardiness, also known as just-in-time, aims to find an optimal schedule that minimizes the total cost of earliness and tardiness for a given set of jobs by determining work allocations that reach the completion of jobs as close as possible to their due dates. Here we consider that the earliness and tardiness penalties are equal for the same job; this is, one period of earliness on a job is as costly as one period of tardiness on the same job. The nomenclature for the decision variables of the model is given in Table 6.20.

Similarly to the implementation of the BPS model to the TWT problem, this model uses the binary variables y_{it} to track the completion time of jobs while the

Table 6.19: APS Extended Testing - Sparse Release Dates on 40 Jobs(40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	19	1964	3946	4661	0.00	2.87	5.06
PS	20	3623	4031	4417	2.34	3.56	5.57

Table 6.20: Decision Variables of the BPS Model Applied to the TWET Problem

x_{it}	Fraction of a job i assigned during regular time at period t
y_{it}	Binary variable: 1 if job i finishes on period t, 0 otherwise
f_i	Finish date for job i
e_i	Number of early periods on job i
l_i	Number of late periods on job i

continuous variables determine the amount of allocations of regular capacity x_{it} that are necessary to complete the jobs. Discrete variables are used to evaluate the finish date f_i as well as the number of early or late periods e_i or l_i respectively. The earliness or tardiness cost for job i is captured by a single cost rate per period w_i as in $w_i(e_i + l_i) = w_i \times max\{d_i - f_i, f_i - d_i\}$. Consequently, the mathematical model is as follows:

$$\text{minimize} \sum_{i=1}^{n} w_i(e_i + l_i) \tag{6.55}$$

subject to:

$$\sum_{t=r_i}^{T} y_{it} = 1, \qquad \forall i \in I \tag{6.56}$$

$$f_i = \sum_{t=1}^{T} t y_{it}, \qquad \forall i \in I \tag{6.57}$$

$$l_i \geq f_i - d_i, \qquad \forall i \in I \tag{6.58}$$

$$e_i \geq d_i - f_i, \qquad \forall i \in I \tag{6.59}$$

$$\sum_{i=1}^{n} x_{it} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.60}$$

$$\sum_{t=r_i}^{T} x_{it} = p_i, \qquad \forall i \in I \tag{6.61}$$

$$\sum_{t=r_i}^{\gamma} x_{it} \geq p_i y_{i\gamma}, \qquad \forall i \in I,\ \forall \gamma \in \{r_i, \ldots, T\} \tag{6.62}$$

$$x_{it} \in [0, 1]\,; \quad f_i,\ l_i \in \mathbb{Z}^+; \quad y_{it} \in \{0, 1\} \tag{6.63}$$

The objective function given in (6.55) minimizes the total cost of weighted earliness and tardiness. Similarly to all the previous BPS models, the weight for job i, w_i, maps the unit cost of earliness or tardiness for that job. The first set of constraints (6.56) ensure that each job is finished by the end of the planning horizon. The second set of constraints (6.57) capture the completion times of each one of the jobs. Given the completion times, constraints (6.58) and (6.59) determine the number of early or tardy periods for each one of the jobs. Constraints (6.60) enforce the capacity limit designated as α_x. Constraints (6.61) allocates all the work required p_i to complete a job across time periods following its release date, and (6.62) ensures that jobs are not

completed before all required work is done. Solution to the above model provides an optimal preemptive schedule to minimize the earliness and tardiness for a given set of jobs with different release dates.

6.4.2 The APS Model for TWET

While maintaining all the same model structure used in the previous APS models, the implementation of the APS model on the TWET problem requires a few changes to incorporate earliness measurements into the model. All the variables in the model remain the same for the most part while the variable R_{jit} is introduced to capture the number of early periods on a job i belonging to customer j. Since the aggregate structure of the model does not provide a direct measure for R_{jit}, we included two intermediate integer variables V_{jit} and K_{jit} intended to capture the number of periods with work done in advance (i.e. inventory) and the number of periods with assigned capacity respectively. The difference between these two variables lead to the earliness measure. The complete list of decision variables for this model are listed in Table 6.21 and the mathematical model is next.

Table 6.21: Decision Variables of the APS Model Applied to the TWET Problem

X_{jit}	Percent of work assigned to job i in cluster j at period t
K_{jit}	Periods in which job i in cluster j is using work capacity X_{jit}
E_{jit}	Percent of work completed by the deadline for job i in cluster j in period t
V_{jit}	Periods in which job i in cluster j has work completed E_{jit}
B_{jit}	Percent of unfulfilled work after the deadline for job i in cluster j in period t
R_{jit}	Early periods corresponding to job i in cluster j
L_{jit}	Late periods corresponding to job i in cluster j

$$\text{minimize} \sum_{j\in J}\sum_{i\in M_j}\sum_{t=1}^{T} \overline{w}_j(R_{jit}+L_{jit}) \tag{6.64}$$

subject to:

$$B_{ji1} - E_{ji1} = D_{ji1} - X_{ji1}, \qquad \forall j \in J, \forall i \in M_j \tag{6.65}$$

$$B_{jit} - E_{jit} = D_{jit} + B_{jit-1} - X_{jit} - E_{jit-1}, \quad \forall j, i; \forall t \in \{2, 3, ., T\} \tag{6.66}$$

$$\sum_{j\in J}\sum_{i\in M_j} P_{ji}X_{jit} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.67}$$

$$\sum_{t\in T} X_{jit} \leq 1, \qquad \forall j \in J; \forall\ i \in M_j \tag{6.68}$$

$$\sum_{t=1}^{r_{ji}-1} X_{jit} \leq 0, \qquad \forall j \in J, \forall i \in M_j \tag{6.69}$$

$$L_{jit} \geq B_{jit}, \qquad \forall j \in J, \forall\ t \in \{1, 2, \ldots, T\} \tag{6.70}$$

$$V_{jit} \geq E_{jit}, \qquad \forall j \in J, \forall\ t \in \{1, 2, \ldots, T\} \tag{6.71}$$

$$K_{jit} \geq X_{jit}, \qquad \forall j \in J, \forall\ t \in \{1, 2, \ldots, T\} \tag{6.72}$$

$$R_{jit} = V_{jit} - K_{jit}, \qquad \forall j \in J, \forall\ t \in \{1, 2, \ldots, T\} \tag{6.73}$$

$$X_{jit},\ B_{jit},\ E_{jit} \in [0, 1], \qquad \forall j \in J, \forall t \in \{1, 2, \ldots, T\} \tag{6.74}$$

$$L_{jit}, R_{jit}, K_{jit}, V_{jit} \in \mathbb{Z}^+, \qquad \forall j \in J, \forall t \in \{1, 2, \ldots, T\} \tag{6.75}$$

Similarly to the previous APS models, the first two constraints represent the work balance equations which capture the percent of work done after the release of each job in each cluster. Constraint (6.67) enforce the capacity limit. Constraint (6.68) ensures that no job is assigned more work than what is required to complete 100% of the work. Constraint (6.69) ensures that work is not assigned to a job on periods prior to its release date. Constraint (6.70) captures each one of the periods in which a job is late, similarly constraints (6.71)and (6.72) capture each one of the periods

in which a job has work done in advance and assigned work respectively. Solution to the above model provides a feasible optimal preemptive schedule for the TWET problem.

6.4.3 Computational Experiments

We test the APS and BPS modeling approaches as they are implemented to solve the TWC problem using the same methodology used during the testing of the models on the TWT problem - Section 6.2.3. Here we perform the same two sets of numerical experiments and solve the same instances as in Section 6.2; keeping the same format we test first the performance of both models under five different problem sizes (10, 20, 40, 80, and 160 jobs) followed by tests to evaluate the effect of the arrangement of release and due dates on the APS model. The results of these tests are as follows.

6.4.4 Computational Results

Results from solving 100 instances (20 for each problem size) on each model (200 runs) show that the BPS model was able to find the optimal solution in 20% of the instances (20 of the 100 instances) while the APS model produced optimal solutions in 40% (40 instances out of 100). Figure 6.3 illustrates these observations in more detail. As expected, in both cases, the number of optimally solved instances decreases as the problem size (*i.e.*, the number of jobs) increases.

While the APS model reported optimal solutions for all instances of the smallest problem size (10 jobs), a large portion of these instances did not reach optimality with the BPS model within the preset time limit. The gap between the two models in this

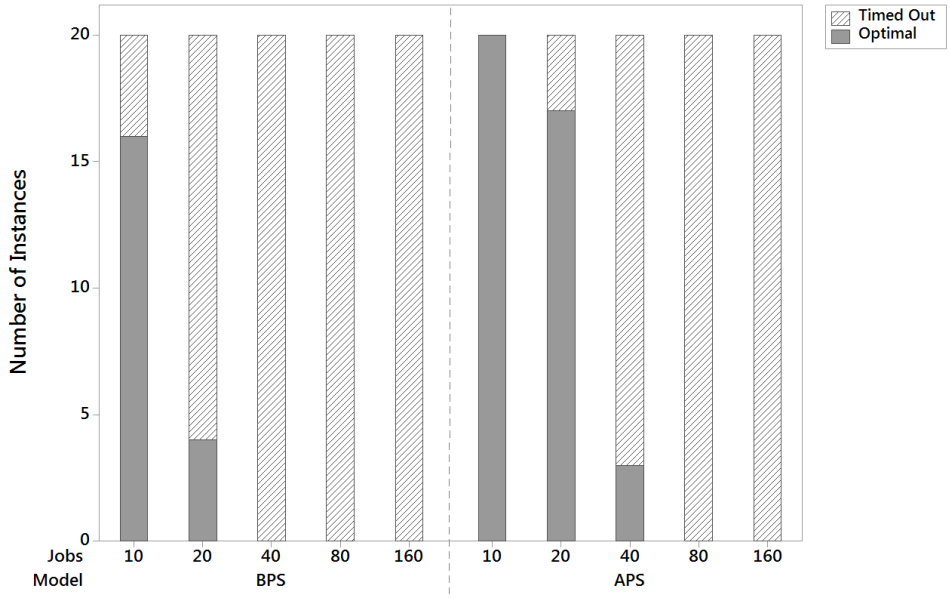

Figure 6.3: Count of optimal and timed-out instances under the TWET problem

respect is also noticeable with 20 and 40 jobs. The BPS model provided 4 optimal solutions for instances with 20 jobs, and no optimal solutions for the instances 40 jobs or larger. The APS model, on the contrary, was able to reach optimality on most of the instances with 20 jobs (17 out of 20), and 3 of the instances with 40 jobs. As we compare the computational time performances and optimality gaps between both models we see a significant difference. In terms of computational time performances the APS model we observe that even with relatively small size problems (*i.e.*, 10 jobs), while it takes on average 18 minutes for the BPS model to reach at optimality, the APS achieves the same result in a matter of 12 seconds. As the problem size increases, the convergence performance degrades significantly for the BPS model. The average optimality gap quickly rises to 19.5% for 40 jobs and continues to increase with larger sizes with a worst case performance of 85.5% gap on one of the instances wiht 160 jobs. On the contrary, the optimality gap with the APS model is well below 5% in

the case of 40 jobs and it increases up to about 32% for 160 jobs. Numeric details on the comparison is summarized in tables 6.22 and 6.23.

Table 6.22: Results for the BPS Model on the TWET Problem (100 instances)

	BPS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	4	0.05	1093	3607	0.00	0.00	0.00
20	16	3.28	2941	3604	0.00	1.29	8.17
40	20	3600	3605	3635	0.43	19.5	50.5
80	20	3600	3611	3648	44.7	63.8	82.9
160	20	3611	3758	3968	75.25	81.66	85.54

Table 6.23: Results for the APS Model on the TWET Problem (100 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.093	11.91	149.8	0.00	0.00	0.00
20	3	3.040	840.2	3991	0.00	0.50	4.18
40	17	583.6	3587	4255	0.00	3.06	6.21
80	20	3601	3638	3676	1.97	8.21	18.1
160	20	3606	3663	4076	2.46	31.95	97.16

6.4.4.1 Extended Testing for the APS Model

Similarly to the extended testing performed on the APS model under the TWC and TWT problems, numerical experiments show how the model performance under the TWET is also affected by the arrangement of release and due dates in the schedule; Similar to the TWT, here the two levels of problem sizes (20 and 40 Jobs) consistently show that instance complexity increases also as a function of the relative arrangement

of the release dates and due dates. Across the release date arrangements it is observed that the number of instances solved optimally decreases when the jobs are released in close proximity to each other; the effect of condensed release dates is noted more dramatically under the group of 20 jobs where the total number of optimally solved instances goes from 30 to 11; as expected, the computational time also change as the release dates are arranged closer together, and on average the computational time for 20 jobs goes from 1113 seconds to 3041. In the case of 40 jobs this trend is not observed as the larger number of timed out instances forced most of the computational times to be near the time out limit (3600 Seconds). In terms of the optimality gap the results for the 20-jobs group are consistent with the results on the number of time outs and computational times; here the average gap grows from 0.72% to 1.34% as the release dates are more compact. A not so consistent and conclusive pattern is observed in the optimality gaps for the 40-jobs group where the optimality gaps change, by contrast, from 1.89% to 3.02% as the release dates are spread apart.

Across the due date arrangements a similar pattern is observed for the 20-jobs group although in a less significant way. The number of instances that timed out, the computational times, and the optimality gaps reflect a consistent change in their magnitude in the direction of more dense arrangements of release dates and due dates; this is, the number of timed out instances grows from 18 to 21, the computational time changes from an average of 2031 seconds to 2123, and the optimality gaps go from 0.92% to 1.14% all as due date slack is restricted within a small constant range; for 40-jobs, again, the results are not conclusive; here the total number of timed out instances, the average computational times and the optimality gaps do not reflect a

significant change as the due date slack changes. A detailed account of the specific results for each test group is summarized in tables 6.24 through 6.27.

Table 6.24: APS Extended Testing - Condensed Release Dates on 20 Jobs (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	14	91.62	2859	3861	0.00	1.33	3.39
PS	15	351.5	3222	4145	0.00	1.34	3.97

Table 6.25: APS Extended Testing - Sparse Release Dates on 20 Jobs (40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	7	4.830	1387	3981	0.00	0.95	3.71
PS	3	3.040	840.2	3991	0.00	0.50	4.18

Table 6.26: APS Extended Testing - Condensed Release Dates on 40 Jobs (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	20	3614	3905	4126	1.14	1.83	2.56
PS	20	3604	3863	4129	1.39	1.96	2.67

Table 6.27: APS Extended Testing - Sparse Release Dates on 40 Jobs (40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	16	491.2	3365	4197	0.00	2.97	9.76
PS	17	583.6	3587	4255	0.00	3.06	6.21

6.5 The Total Weighted Number of Tardy Jobs Problem (TWNTJ)

6.5.1 The BPS Model for TWNTJ

The optimization models for the total weighted number of tardy jobs differently from the TWT model, as its name implies, are only concerned with the jobs that are tardy regardless of how large the tardiness is; consequently, the BPS model does not require a tardiness variable l_{it}; instead it requires an additional binary variable U_i indicating if a job i is late or not. The nomenclature for the decision variables of the model is given in Table 6.28.

Table 6.28: Decision Variables of the BPS Model Applied to the TWNTJ Problem

x_{it}	Fraction of a job i assigned during regular time at period t
y_{it}	Binary variable: 1 if job i finishes on period t, 0 otherwise
U_i	Binary variable: 1 if job i is late, 0 otherwise
f_i	Finish date for job i

Similarly to the previous models this model involves binary, continuous, and discrete decision variables. The two binary variables y_{it} and U_i track respectively the completion time of jobs, and whether jobs are late or not. The continuous variables determine the amount of allocations of regular capacity x_{it} that are necessary to com-

plete the jobs while the discrete variables track the finish dates f_i . The mathematical model is as follows:

$$\text{minimize} \sum_{i=1}^{n} w_i U_i \tag{6.76}$$

subject to:

$$TU_i \geq f_i - d_i, \qquad \forall i \in I \tag{6.77}$$

$$\sum_{t=r_i}^{T} y_{it} = 1, \qquad \forall i \in I \tag{6.78}$$

$$f_i = \sum_{t=1}^{T} t y_{it}, \qquad \forall i \in I \tag{6.79}$$

$$\sum_{i=1}^{n} x_{it} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.80}$$

$$\sum_{t=r_i}^{T} x_{it} = p_i, \qquad \forall i \in I \tag{6.81}$$

$$\sum_{t=r_i}^{\gamma} x_{it} \geq p_i y_{i\gamma}, \qquad \forall i \in I,\ \forall \gamma \in \{r_i, \ldots, T\} \tag{6.82}$$

$$x_{it} \in [0, 1]\,; \quad f_i \in \mathbb{Z}^+; \quad y_{it}, U_i \in \{0, 1\} \tag{6.83}$$

The objective function given in (6.76) minimizes the total cost of tardy jobs weighted by different amounts w_i where the weight for job i, w_i, maps the cost incurred when job i is late. The first set of constraints (6.77) identifies when a particular job is late; this is done via the helping parameter T which represents the total number of periods in the planning horizon. The following constraints (6.78) ensures that each job is finished by the end of the planning horizon, and constraints (6.79) capture the completion times of each one of the jobs. Constraints (6.80) enforces the

capacity limit designated as α_x. Constraint (6.81) allocates all the work required p_i to complete a job across time periods following its release date, and (6.82) ensures that jobs are not completed before all required work is done. Solution to the above model provides an optimal preemptive schedule to minimize the total number of tardy jobs weighted by different cost amounts.

6.5.2 The APS Model for TWNTJ

The implementation of the APS model on the TWNTJ problem although very similar to the TWT problem requires a few constraint changes to incorporate the count of tardy jobs into the model. Similarly to the BPS model, the amount of tardiness is not relevant; therefore, the variable L_{jit} is replaced by a binary variable U_{ji} that tracks whether a job is late or not. The rest of the variables in the model remain the same. The complete list of decision variables for this model are listed in Table 6.21.

Table 6.29: Decision Variables of the APS Model Applied to the TWNTJ Problem

X_{jit}	Percent of work assigned to job i in cluster j at period t
E_{jit}	Percent of work completed by the deadline for job i in cluster j in period t
B_{jit}	Percent of unfulfilled work after the deadline for job i in cluster j in period t
U_{ji}	1 if job i in cluster j is late, 0 otherwise

$$\text{minimize} \sum_{j \in J} \sum_{i \in M_j} \overline{w}_j U_{ji} \tag{6.84}$$

subject to:

$$TU_{ji} \geq \sum_{t \in T} B_{jit}, \qquad \forall j \in J, \forall\ i \in M_j \tag{6.85}$$

$$B_{ji1} - E_{ji1} = D_{ji1} - X_{ji1}, \qquad \forall j \in J, \forall i \in M_j \tag{6.86}$$

$$B_{jit} - E_{jit} = D_{jit} + B_{jit-1} - X_{jit} - E_{jit-1}, \quad \forall j, i; \forall t \in \{2, 3, ., T\} \tag{6.87}$$

$$\sum_{j \in J} \sum_{i \in M_j} P_{ji} X_{jit} \leq \alpha_x, \qquad \forall t \in \{1, 2, \ldots, T\} \tag{6.88}$$

$$\sum_{t \in T} X_{jit} = 1, \qquad \forall j \in J; \forall\ i \in M_j \tag{6.89}$$

$$\sum_{t=1}^{r_{ji}-1} X_{jit} \leq 0, \qquad \forall j \in J, \forall i \in M_j \tag{6.90}$$

$$X_{jit},\ B_{jit},\ E_{jit} \in [0, 1], \qquad \forall j \in J \forall t \in \{1, 2, \ldots, T\} \tag{6.91}$$

$$U_{ji} \in \{0, 1\} \qquad \forall j \in J, \forall i \in M_j \tag{6.92}$$

The first constraint (6.85) tracks via the binary variable U_{ji} all the jobs that are being completed after the due date (i.e. jobs that contain pending work $B_{jit} > 0$); again, this is done via a helping parameter T representing the total number of periods in the planning horizon. The two following constraints (6.86, 6.87) represent the work balance equations; constraints (6.88) enforce the capacity limit. Constraint (6.89) ensures that no job is assigned more work than what is required to complete 100% of the work. Constraint (6.90) ensures that work is not assigned to a job on periods prior to its release date. Solution to the above model provides a feasible optimal preemptive schedule to minimize the total number of weighted tardy jobs.

6.5.3 Computational Experiments

The testing of the APS and BPS modeling approaches as they are implemented to solve the TWNTJ problem follows the same methodology used during the testing of the TWT problem - Section 6.2.3. For experimental consistency throughout the research, regardless of the relevance of the results, here we perform and discuss the same two sets of numerical experiments as in Section 6.2.

6.5.4 Computational Results

A total of 100 instances (20 for each problem size) on each model (200 runs) show via numerical analysis that the number of instances that were solved optimally by each model was again significantly different; however, the total number of instances solved optimally was significantly higher than the number of optimal instances reported for the TWT, TWC and TWET problems. The BPS model was able to find the optimal solution in 50% of the instances (50 of the 100 instances) while the APS model produced optimal solutions in 88% (88 instances out of 100). Figure 6.4 illustrates these observations in more detail. As expected, in both cases, the number of optimally solved instances decreases as the problem size (*i.e.*, the number of jobs) increases.

While the APS model reported optimal solutions for all instances of problem size of 80 jobs or less, the BPS model did not find the optimal solution on some portion of these instances. The gap between the two models in this respect is also apparent with 160 jobs. The BPS model provided no optimal solutions for instances with 80 jobs or higher whereas the APS model was able to reach optimality on some of the instances with 160 jobs. Results are summarized in tables 6.30 and 6.31.

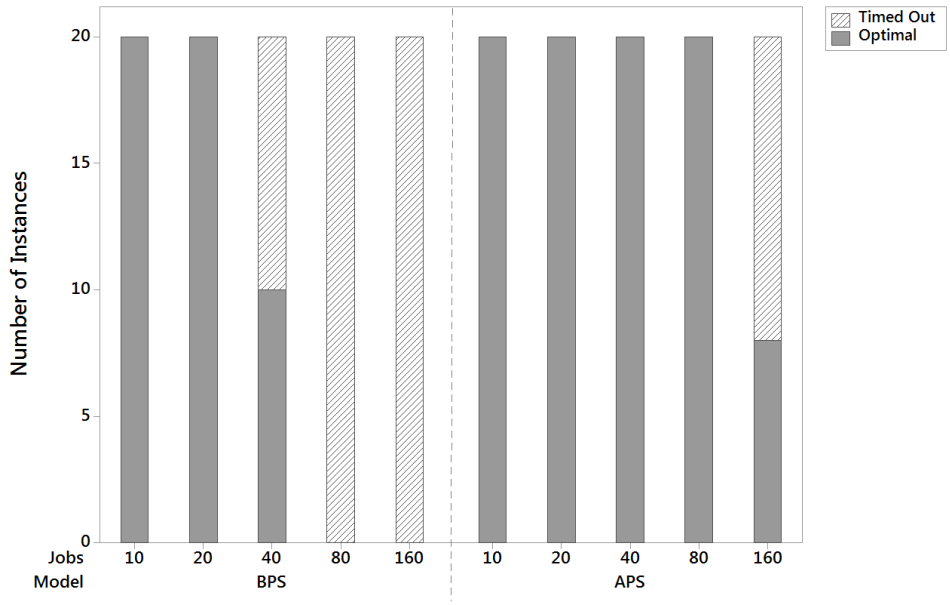

Figure 6.4: Count of optimal and timed-out instances under the TWNTJ problem

In terms of computational time, we observe that even with 20-jobs size problems the BPS model reaches optimality after about 17 seconds on average while the APS achieves the same result within a fraction of a second on average. As the problem size increases, the convergence performance on the BPS model degrades although not as dramatically as in the previous tardiness related problems, and the average optimality gap goes to up to 8.63% for 160 jobs with a worst case performance of 14.02% gap. In the case of the APS model, the optimality gap is not much different with an average value under 8% and a worst case performance of 54.93% on 160 jobs. In terms of the maximum values reported for the optimality gap with 160 jobs, the BPS model appears to produce significantly lower values than the APS model (14.02% vs. 54.93% respectively); this data artifact is the result of the lack of feasible solutions from the BPS runs which resulted in only 4 instances with feasible integer solutions and 16 instances for which the time limit was surpassed without the solver reaching any integer feasible solution.

Table 6.30: Results for the BPS Model on the TWNTJ Problem (100 instances)

	BPS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.05	0.96	2.28	0.00	0.00	0.00
20	0	1.06	17.30	132.2	0.00	0.00	0.00
40	10	18.51	2111	3995	0.00	0.49	5.81
80	20	3602	3683	4267	0.00	1.38	8.76
160	20*	3620	3624	3632	4.68	8.63	14.02

*Only 4 out of the 20 instances reached a feasible integer solution

Table 6.31: Results for the APS Model on the TWNTJ Problem (100 instances)

	APS						
	Computational Time (sec)				Relative Gap (%)		
Jobs	TO	Min	Avg	Max	Min	Avg	Max
10	0	0.015	0.116	0.281	0.00	0.00	0.00
20	0	0.078	0.920	2.888	0.00	0.00	0.00
40	0	2.263	9.436	32.17	0.00	0.00	0.00
80	0	13.04	315.2	2671	0.00	0.00	0.00
160	12	554.8	2475	3636	0.00	7.89	54.93

6.5.4.1 Extended Testing for the APS Model

As observed in the previous section, the performance of the models under the TWNTJ is significantly better than what we observed on other problems; consequently, the numerical experiments focused on the APS model for the two levels of problem sizes (20 and 40 Jobs) consistently show a larger number of optimally solved instances than in previous experiments. In this case, the availability of useful data on the performance of the APS model as it is affected by the arrangement of release and due dates is somewhat hindered by the better performance of the model; this

is, across the release date arrangements it is observed that the number of instances that timed out in the 20 and 40 job groups is virtually zero, and this is also reflected on the optimality gaps where all the values were zero. With respect to the computational time and the release date arrangement, the data also shows no significant differences where for 20 and 40 jobs the average amount of time required to reach optimal solutions was virtually unchanged by the different arrangements of release dates.

Across the due date arrangements a similar situation is observed; the number of instances solved optimally as well as the computational times and optimality gaps do not offer a clear view of the effect produced by the arrangement of the due dates. Overall, the impact of the release and due date slack arrangements under the TWNTJ problem did not provide major insights on the sensitivity of the model to these factors. A summary of the results for each test group is summarized in tables 6.32 through 6.35.

Table 6.32: APS Extended Testing - Condensed Release Dates on 20 Jobs (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	0*	0.400	0.811	1.970	0.00	0.00	0.00
PS	0	0.437	1.091	2.930	0.00	0.00	0.00

* 3 out of the 20 instances were infeasible

6.6 Analytical Comparison of Models

From a practical perspective the overall superior performance of the APS modeling approach has been shown via numerical testing. In this part of our discussion we

Table 6.33: APS Extended Testing - Sparse Release Dates on 20 Jobs (40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	0	0.090	0.903	1.995	0.00	0.00	0.00
PS	0	0.078	0.920	2.888	0.00	0.00	0.00

Table 6.34: APS Extended Testing - Condensed Release Dates on 40 Jobs (40 instances)

	Condensed Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	0*	1.627	9.389	29.79	0.00	0.00	0.00
PS	0	4.773	10.16	23.41	0.00	0.00	0.00

* 1 out of the 20 instances was infeasible

continue the comparison of the presented approaches and models but this time under a more analytic umbrella. The sections ahead are targeted to examine two different aspects that support the claims made after the numerical testing; the first section shows that the BPS as well as the APS modeling approaches as applied to the different tardiness related measures are equivalent at providing the same solution to the same problem; the second section takes a look at the solution space of both models and shows how the differences in the size of the solution spaces correlate to the differences in performance.

6.6.1 Model Equivalency

A closer look at all the model implementations in sections 6.2 through 6.5 shows that for each modeling approach (APS or BPS) the core modeling structure remains the same regardless of the tardiness measure being addressed; this is, for the BPS

Table 6.35: APS Extended Testing - Sparse Release Dates on 40 Jobs (40 instances)

	Sparse Release Dates						
	Computational Time (sec)				Relative Gap (%)		
Slack	TO	Min	Avg	Max	Min	Avg	Max
CS	0	1.346	16.03	111.9	0.00	0.00	0.00
PS	0	2.263	9.436	32.17	0.00	0.00	0.00

models as they are applied to the TWT, TWC, TWET and TWNTJ the parameters (r, d, x, α_x and p) constitute a set of controlling constraints that remain invariable from problem to problem; similarly the APS models in all tardiness measures uses the same work balance equations and controlling constraints throughout all problems; furthermore, a closer look between the APS and BPS model implementations on any one of the problems reveals that each of the core set of constraints in the BPS model has a direct equivalent within the core set of constraints in the APS model.

In general, for all tardiness related measures it is possible to see that the APS and BPS models are equivalent when comparing their core set of constraints; however, such equivalency is not as straight forward when comparing their objective functions; therefore, model equivalence in this context is mainly concerned with the equivalency between the representations of the tardiness measures in each model (*i.e.* the objective functions in each model). With this in mind, here we show model equivalency from an objective function perspective; to this end we use the TWT problem as our core problem to show equivalency between its BPS and APS objective functions; then we show that these results are also applicable to show the equivalency between the models in all the remaining problems.

From evaluating the terms in the objective functions of the models in the TWT problem, we can see that the tardiness variables (l_i and L_{jit}) don not map directly

from one model to the other. While in the BPS model l_i represents the actual number of tardy periods of job i, the variable L_{jit} in the APS model only tracks the individual periods in which a job i in customer j is actually late.

$$BPS: \qquad \sum_{i=1}^{n} w_i l_i$$

$$APS: \qquad \sum_{j\in J}\sum_{i\in I}\sum_{t=1}^{T} \overline{w}_j L_{jit}$$

For both objective functions to match one-to-one the tardiness expressions $\sum_{t=1}^{T} L_{jit}$ and l_i must be equivalent as shown in expression (6.93).

$$l_i = \sum_{t=1}^{T} L_{jit} \tag{6.93}$$

In order to validate the equivalence in (6.93) the summation of L_{jit} over t must amount to the same tardiness represented by $l_i = f_i - d_i$ in the BPS model; therefore,

$$\sum_{t=1}^{T} L_{jit} = f_i - d_i \tag{6.94}$$

Since $L_{jit} \geq B_{jit}$ as listed in constraints (6.15) of the APS model, it is clear that each tardy period L_{jit} is tied to periods in which there is pending work B_{jit}; furthermore, B_{jit} is constrained to satisfy the work balance equation $B_{jit} - E_{jit} = D_{jit} + B_{jit-1} - X_{jit} - E_{jit-1}$ where the variables E_{jit} cannot be greater than zero on periods t where $B_{jit} \geq 0$ and E_{jit} cannot be greater than zero on periods $t \geq argmax_{t\in\{1,2,\ldots T\}}(D_{jit})$. Being the precise period when a job is due t_d represented by:

$$t_d = argmax_{t\in\{1,2,\ldots T\}}(D_{jit}) \tag{6.95}$$

and the value of B on the due date B_{jit_d}

$$B_{jit_d} = D_{jit_d} - (X_{jit_d} + E_{jit_d-1}) \tag{6.96}$$

Where $B_{jit_d} \in [0, D_{jit_d}]$ and $X_{jit_d} \in [0, D_{jt_d}]$.

In light of the above premises (expressions 6.95, 6.96) we can express the total tardiness of a job i in j as

$$\sum_{t=1}^{T} L_{jit} + t_d - t_d = \sum_{t=1}^{T} \lceil B_{jit} \rceil + t_d - t_d \tag{6.97}$$

Where on the left side $\sum_{t=1}^{T} L_{jit} + t_d$ is equivalent to the finish date which we label as t_f and on the right side can be broken down into the summation of B values before the due period ($t \leq t_d - 1$), between the due period and the finish period ($t_d \leq t \leq t_f - 1$), and after the finish period ($t \geq t_f$)

$$\sum_{t=1}^{T} B_{jit} = \sum_{t=1}^{t_d-1} B_{jit} + \sum_{t=t_d}^{t_f-1} B_{jit} + \sum_{t=t_f}^{T} B_{jit} \tag{6.98}$$

Here the first and last term on the right side are equivalent to zero and the middle term represents the equivalent non-integer expression for tardiness where $\sum_{t=t_d}^{t_f-1} B_{jit} + t_d$ represents the non-integer expression for the finish date, and showing that the amount of tardiness is directly equivalent from one model to another as in (6.99)

$$l_i = \sum_{t=t_d}^{t_f-1} \lceil B_{jit} \rceil + t_d - t_d \tag{6.99}$$

As the tardiness related measures studied in this document are derived from the TWT problem, the equivalency between the APS and BPS models in the TWT problem can be easily used to derive the equivalency of the objective functions in the

remaining problems; this is, in the case of the TWC problem the equivalency between the tardiness variables $l_i = \sum_{t=1}^{T} L_{jit}$ remain unchanged as these same variables are used to represent completion times C_i and C_{jit} when all jobs are defined as tardy; in the case of the TWET a new variable is added to each model to account for the earliness however it can be seen that this does not change the equivalency as in (6.100)

$$(l_i + e_i) = \sum_{t=1}^{T} (L_{jit} + R_{jit}) \tag{6.100}$$

Finally, it can be seen how the objective functions in the TWNTJ directly follow equivalency as a result of the new variable U where the specific amounts of lateness are not relevant and equivalency is based simply on the presence or absence of lateness which as shown above are also equivalent.

6.6.2 Solution Space and Model Performance

The size of the solution space as an indicator of the computational burden is also a way of estimating the overall performance of a model; in particular, here we show that the APS model possesses a lower computational burden than the BPS model which results in faster solutions. By solution space, in this context, we mean the number of possible combinations that the independent variables can take in order to generate a single feasible solution. We use the TWT models as basis for comparison, and since the model structure does not change for the different tardiness related measures, the results from this section can be extended to all the other tardiness related problems (TWET, TWC and TWNTJ). We initiate this discussion using the BPS model to compute the size of its solution space; in this particular case a single feasible solution with a cost ψ equal to:

$$\psi = \sum_{i=1}^{n} w_i l_i \tag{6.101}$$

Being T the total number of periods in the planning horizon and n the total number of jobs in the set of jobs I; the model must compute the lateness for each one of the n jobs (*i.e.* $l_i : i \in \{1, 2, \ldots, n\}$) where $l_i = f_i - d_i$ and $f_i = \sum_{t=1}^{T} t y_{it}$ subject to $\sum_{t=1}^{T} y_{it} = 1, \forall i \in I$. Considering this, to produce a single feasible value for l_i the binary variable y_{it} must try from a combination of binary options in the elements of a vector $\hat{t} =< a_1, a_2, \ldots, a_T >$ where each element $a_t \in \{0, 1\}$ for $t \in \{1, 2, \ldots, T\}$. Given T binary digits in each job there are only T feasible options for y_{it} to take the value of 1 only once within $\hat{t}$; for an instance, if $T = 3$ there will only be three possible options for y_{it} to take the value of 1. this is illustrated as follows in figure 6.5

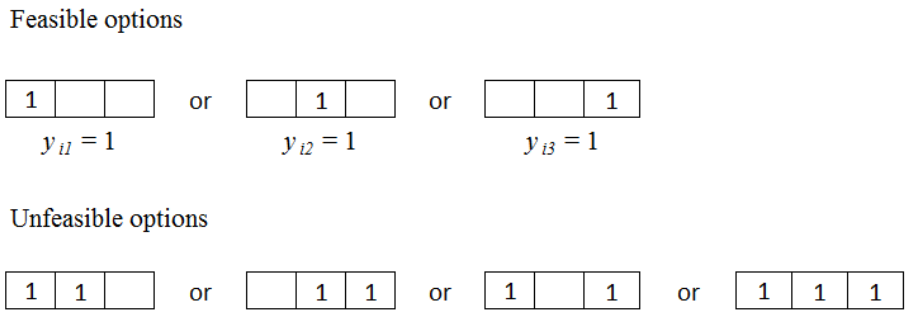

Figure 6.5: Sample feasible and unfeasible options for y_{it}

Here the total number of possible non-zero value options S_T (feasible and unfeasible) for an independent variable such as y_{it} on a single job is:

$$S_T = T + \frac{T!}{2!(T-2)!} + \frac{T!}{3!(T-3)!} + \cdots + \frac{T!}{T!(T-T)!}$$

$$S_T = \sum_{k=1}^{T} \frac{T!}{k!(T-k)!}$$

$$S_T = \sum_{k=1}^{T} \binom{T}{k}$$

In addition to y_{it}, the BPS model also contains x_{it} as an independent variable; therefore using a similar rationale we can see that there are also S_T possible non-zero value options for x_{it}. By combining the available options for all the independent variables in the BPS model we can see that for every single job the solution sub-space in the BPS model SS_{job} is given by :

$$SS_{job} = S_T \times S_T = (S_T)^2 \tag{6.102}$$

and for a group of n jobs the BPS model solution space SS_{BPS} is:

$$SS_{BPS} = n(S_T)^2 \tag{6.103}$$

Analogously, by looking at the model variables in the APS model, it can be noted that only X_{jit} represents an independent variable. Following the same logic from the binary model the number of non-zero value options for X_{jit} is also given by S_T, and since there is only one independent variable in the APS model we can conclude that the total solution space for the APS model is given by SS_{APS}

$$SS_{APS} = n(S_T) \tag{6.104}$$

which indicates that the solution space for the BPS model is S_T times larger than the one for the APS; furthermore, we can conclude that since the variables X_{jit} map directly to the variables x_{it} in the BPS model, the APS solution space is contained in the BPS solution space:

$$SS_{APS} \subset SS_{BPS} \qquad \square \tag{6.105}$$

6.7 Conclusions

We propose two modeling approaches that can be used to solve a variety of tardiness related problems on a single machine layout where preemption is allowed. Following the conventional modeling approach, we first build a model using binary variables that identify completion times of a finite set of jobs with varying release times, due dates, and processing times - we refer to this model as the Binary Preemptive Scheduling (BPS) model. The second model, which we call the Aggregate Preemptive Scheduling (APS) model, employs adopts the aggregate planning approach while binary variables are eliminated producing more efficient and advanced models.

In order to test and compare both models for computational efficiency, we design and perform two sets of numerical tests. First we determine and compare the computational performance of both models with respect to the number of jobs for all tardiness related measures. Here we observe that the BPS model was outperformed by the APS model in all the different tardiness measures. Second, we explore the

performance and sensitivity of the APS models to different arrangements of release and due dates, again, for all tardiness related measures; through the extended testing on the APS model, a pattern is noticed where effect of condensed release dates on 40 jobs produced inconclusive results. Aside from the TWNTJ problem where most of the optimality gaps are zero (optimal), the TWT, TWC and TWET problems show an increase in the relative gap as the release dates were spread apart (*i.e.* relaxed). From an empirical perspective we can only speculate that the difference in the behavior of the optimality gap for 20 and 40 jobs implies that a strong interaction is present between the range used for the release dates and the instance size; we know that if the number of jobs in an instance increases but their release dates stays constrained to a narrow range the problem complexity is likely to be reduced; as n tends to infinity and the release date range stays constant, the problem tends to be trivial where a polynomial time solution can be obtained by scheduling the jobs in non-decreasing order of their weights. Aside from the explanations we can provide at this moment we recognize that more research is required in order to fully understand the interaction between problem size and release dates. Overall, results of our analysis demonstrate the strong computational performance and convergence of the APS model and its patent dominance over the BPS model. The study reveals that APS model is a promising tool for generating optimal or near-optimal schedules and capacity allocation plans reasonably quick.

CHAPTER 7

Concluding Remarks

We present three alternative methodologies for the solution of tardiness related problems: heuristics, conventional modeling and advanced modeling approaches each one with a distinctive performance and application. We show how heuristic approaches represent a quick and fairly accurate alternative to the conventional modeling approach, and how the advanced modeling approach represents a quicker and more accurate method than the conventional modeling approach for the solution of tardiness related optimization; furthermore, from a practical perspective, by demonstrating the accuracy and efficiency of the advanced modeling approach we offer a single method that can be used in a variety of tardiness related problems for applications with up to 160 jobs. From an academic perspective, here we also contribute to fill an existing void in the literature on preemptive scheduling; we do this by presenting a research that focuses on preemptive problems and uses modeling as the center of the solution approach.

Bibliography

[1] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[2] T. Abdul-Razaq, C. Potts, and L. van Wassenhove, "A survey of algorithms for the single machine total weighted tardiness scheduling problem," *Discrete Applied Mathematics*, vol. 26, no. 2, pp. 235–253, 1990.

[3] C. Koulamas, "The single-machine total tardiness scheduling problem: review and extensions," *European Journal of Operational Research*, vol. 202, no. 1, pp. 1–7, 2010.

[4] J. K. Lenstra, A. Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics*, vol. 1, pp. 343–362, 1977.

[5] R. McNaughton, "Scheduling with deadlines and loss functions," *Management Science*, vol. 6, no. 1, pp. 1–12, 1959.

[6] J. Y. Y. Lin, "Single machine preemptive scheduling with fixed jobs to minimize tardiness related criteria," *European Journal of Operational Research*, 2005.

[7] K. Bulbul, P. Kaminsky, and C. Yano, "Preemption in single machine earliness/tardiness scheduling," *Journal of Scheduling*, vol. 10, pp. 271–292, 2007.

[8] Z. Tian, C. T. Ng, and T. C. E. Cheng, "An o(n 2) algorithm for scheduling equal-length preemptive jobs on a simgle machine to minimize total tardiness," *Journal of Scheduling*, vol. 9, no. 4, pp. 343–364, 2006.

[9] ——, "Preemptive scheduling of jobs with agreeable due date on a single machine to minimize total tardiness," *Operations Research Letters*, vol. 37, pp. 368–374, 2009.

[10] Y. Hendel, N. Runge, and F. Sourd, "The one-machine just-in-time scheduling problem with preemption," *Discrete Optimization*, vol. 6, pp. 10–22, 2009.

[11] M. Batsyn, B. Goldengorin, P. Sukov, and P. M. Pardalos, *Models, Algorithms, and Technologies for Network Analysis*, ser. Springer Proceedings in Mathematics & Statistics. Springer, 2013, vol. 59, ch. Lower and upper bounds for the preemptive single machine scheduling problem with equal processing times, pp. 11–27.

[12] G. Wang, H. Sun, and C. Chu, "Preemptive scheduling with availability constraints to minimize total weighted completion times," *Annals of Operations Research*, vol. 133, pp. 183–192, 2005.

[13] J. Yuan and Y. Lin, "Single machine preemptive scheduling with fixed jobs to minimize tardiness related criteria," *European journal of operational research*, vol. 164, no. 3, pp. 851–855, 2005.

[14] J. M. van den Akker, G. Diepen, and J. A. Hoogeveen, "Minimizing total weighted tardiness on a single machine with release date and equal-length jobs," *Journal of Scheduling*, vol. 13, pp. 561–576, 2010.

[15] P. Brucker and S. Knust. (2009) Complexity results for scheduling problems. http://www.mathematik.uni-osnabrueck.de/research/OR/class.

[16] C. A. Holloway and R. T. Nelson, "Job shop scheduling with due date and overtime capability," *Management Science*, vol. 21, no. 1, pp. 68–78, 1974.

[17] H. Matsuo, "The weighted total tardiness problem with fixed shipping times and overtime utilization," *Operations Research*, vol. 36, no. 2, pp. 293–307, 1988.

[18] R. Daniels and R. Sarin, "Technical note: Single machine scheduling with controllable processing times and number of jobs tardy," *Operations Research*, vol. 37, no. 6, pp. 981–984, 1989.

[19] C. Akkan, "Overtime scheduling: an application in finite-capacity real-time scheduling," *Journal of the Operational Research Society*, pp. 1137–1149, 1996.

[20] B. Yang, J. Geunes, and W. O'Brien, "A heuristic approach for minimizing weighted tardiness and overtime costs in single resource scheduling," *Computers and Operations Research*, vol. 31, pp. 1273–1301, 2004.

[21] N. K. Freeman, J. Mittenthal, and S. H. Melouk, "Parallel-machine scheduling to minimize overtime and waste costs," *IIE Transactions*, vol. 46, pp. 601–618, 2014.

[22] C.-F. Liaw, "A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem," *Computers & Operations Research*, vol. 26, no. 7, pp. 679–693, 1999.

[23] G. Li, "Single machine earliness and tardiness scheduling," *European Journal of Operational Research*, vol. 96, no. 3, pp. 546–558, 1997.

[24] M. Vanhoucke, E. Demeulemeester, and W. Herroelen, "An exact procedure for the resource-constrained weighted earliness–tardiness project scheduling problem," *Annals of Operations Research*, vol. 102, no. 1-4, pp. 179–196, 2001.

[25] C.-J. Liao and C.-C. Cheng, "A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date," *Computers & Industrial Engineering*, vol. 52, no. 4, pp. 404–413, 2007.

[26] C. A. Yano and Y.-D. Kim, "Algorithms for a class of single-machine weighted tardiness and earliness problems," *European Journal of Operational Research*, vol. 52, no. 2, pp. 167–178, 1991.

[27] S. Chand and H. Schneeberger, "Single machine scheduling to minimize weighted earliness subject to no tardy jobs," *European Journal of Operational Research*, vol. 34, no. 2, pp. 221–230, 1988.

[28] G. Wan and B. P.-C. Yen, "Single machine scheduling to minimize total weighted earliness subject to minimal number of tardy jobs," *European Journal of Operational Research*, vol. 195, no. 1, pp. 89–97, 2009.

[29] J. M. Moore, "An n job, one machine sequencing algorithm for minimizing the number of late jobs," *Management science*, vol. 15, no. 1, pp. 102–109, 1968.

[30] T. E. Cheng, C. Ng, and J. Yuan, "Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs," *Theoretical Computer Science*, vol. 362, no. 1, pp. 273–281, 2006.

[31] T. Cheng, Z.-L. Chen, and C.-L. Li, "Single-machine scheduling with trade-off between number of tardy jobs and resource allocation," *Operations Research Letters*, vol. 19, no. 5, pp. 237–242, 1996.

[32] W.-J. Chen, "Minimizing number of tardy jobs on a single machine subject to periodic maintenance," *Omega*, vol. 37, no. 3, pp. 591–599, 2009.

[33] A. Hariri and C. N. Potts, "An algorithm for single machine sequencing with release dates to minimize total weighted completion time," *Discrete Applied Mathematics*, vol. 5, no. 1, pp. 99–109, 1983.

[34] H. Belouadah, M. E. Posner, and C. N. Potts, "Scheduling with release dates on a single machine to minimize total weighted completion time," *Discrete Applied Mathematics*, vol. 36, no. 3, pp. 213–231, 1992.

[35] E. Lawler, "A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness," *Annals of Discrete Mathematics*, vol. 1, pp. 343–362, 1977.

[36] J. Labetoulle, E. Lawler, J. Lenstra, and A. Rinnooy Kan, "Progress in combinatorial optimization," *Academicpress, NewYork*, pp. 245–261, 1984.

[37] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[38] M. R. Garey, R. E. Tarjan, and G. T. Wilfong, "One-processor scheduling with symmetric earliness and tardiness penalties," *Mathematics of Operations Research*, vol. 13, no. 2, pp. 330–348, 1988.

[39] M. Batsyn, B. Goldengorin, P. Pardalos, and P. Sukov, "Online heuristic for the preemptive single machine scheduling problem of minimizing the total weighted completion time," *Optimization Methods and Software*, vol. 5, no. 29, pp. 955–963, 2014.

[40] M. Erkoc and K. Ertogral, "Overhaul planning and exchange scheduling for maintenance services with rotable inventory and limited processing capacity," *Computers & Industrial Engineering*, vol. 98, pp. 30–39, 2016.