SABANCI UNIVERSITY
Orhanlı-Tuzla, 34956 Istanbul, Turkey
Phone: +90 (216) 483-9500
Fax: +90 (216) 483-9550
http://www.sabanciuniv.edu

http://algopt.sabanciuniv.edu/projects
June 26, 2014

# A Strong Preemptive Relaxation for Weighted Tardiness and Earliness/Tardiness Problems on Unrelated Parallel Machines

Halil Şen and Kerem Bülbül
Sabancı University, Manufacturing Systems and Industrial Engineering, Orhanlı-Tuzla, 34956 İstanbul, Turkey.
halilsen@sabanciuniv.edu, bulbul@sabanciuniv.edu

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

*Keywords*: unrelated parallel machines; weighted tardiness; weighted earliness and tardiness; preemptive relaxation; Benders decomposition; transportation problem; lower bound; heuristic.

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

The scheduling literature is often criticized for its emphasis on the single machine environment which is arguably not encountered frequently in today's complex shop floors. However, virtually every scheduling algorithm conceived for multi-stage production systems does either generalize or depend upon the fundamental principles derived from the basic single machine scheduling problems. A similar argument is valid for the parallel machine environment as well. Decomposition algorithms devised for multi-stage systems, such as Lagrangian relaxation, Dantzig-Wolfe reformulation, Benders decomposition, and the shifting bottleneck heuristic, give rise to either single- or parallel machine scheduling subproblems that have to be solved many times in an iterative framework. The ultimate performance of such decomposition approaches depends critically on our ability to solve these subproblems with a high solution quality in short computational times. Moreover, from a theoretical perspective the study of parallel machines is the immediate logical extension of single machine scheduling. For a given partition of the set of jobs over the set of machines, a parallel

machine scheduling problem is just a collection of independent single machine scheduling problems. Therefore, parallel machine scheduling problems are generally regarded as set partitioning problems where the complexity of calculating the cost of a partition depends on the difficulty of the underlying single machine scheduling problem. Motivated by these practical and theoretical considerations, our primary objective in this paper is to devise a fast and effective mathematical programming based heuristic for two fundamental due date related objectives on unrelated parallel machines.

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

The review of the related literature in Section 2 identifies the lack of strong lower bounds as a major impediment to the development of exact algorithms and the performance analysis of heuristics for the TWT and TWET objectives in the parallel machine environment. Shim and Kim (2007a) attack the unweighted version of *Rm-TWT*, and their B&B algorithm does not scale beyond 5 machines and 20 jobs. In their concluding remarks, the authors state that "..., further research is needed if one needs to solve problems of larger or practical sizes. One way may be to develop more effective or tighter lower bounds since the lower bound used in the B&B algorithm suggested in this study does not seem to be very tight." More generally, in their effort to compute strong LP based bounds for a class of parallel machine scheduling problems with additive objectives, van den Akker et al. (1999) observe that "additive objective functions pose a computational challenge because it is difficult to compute strong lower bounds." These comments provide a strong motivation for this study. All promising existing results assume that the machines are identical and often exploit this fact in some way; e.g., by aggregating the machine capacity constraints. Clearly, such approaches do not necessarily extend to or yield similar results for unrelated parallel machines. In this paper, we set out to provide tight lower bounds and near-optimal solutions for the TWT and TWET objectives in the unrelated parallel machine environment. To this end, we propose a new preemptive relaxation that explicitly assigns jobs to specific machines. This preemptive relaxation generalizes and builds upon the success of the related previous studies on the single machine weighted tardiness and weighted earliness/tardiness scheduling problems (Bülbül et al., 2007, Pan and Shi, 2007, Şen and Bülbül, 2012, Sourd and Kedad-Sidhoum, 2003). The resulting lower bound is tight, and perhaps more importantly, the job partition retrieved from the (near-) optimal solution of the preemptive relaxation provides us with sufficient information to construct feasible non-preemptive schedules of high quality for the original problem. That is, we recognize that the main practical difficulty of solving *Rm-TWT* and *Rm-TWET* to (near-) optimality is determining a good job partition, and we directly incorporate this aspect of the problem into our rationale for developing this particular relaxation. Once a job partition is available, we rely on recent advances by Tanaka et al. (2009) and Tanaka and Fujikuma (2012) to solve $m$ independent single machine TWT or TWET problems, respectively, to construct a non-preemptive solution of high quality to the original unrelated parallel machine scheduling problem. The downside of our preemptive relaxation is that it is formulated as a difficult mixed integer linear program. A key contribution of this paper is devising a computationally effective Benders decomposition algorithm that can handle

very large instances of this formulation. Here, the *lazy constraint* generation scheme of IBM ILOG CPLEX (2011) proves instrumental for a successful implementation. Moreover, as we point out in the previous paragraph, both objectives TWT and TWET are tackled successfully by the same algorithm.

In the next section, we review the related literature and put our work into perspective. We introduce and formulate the proposed preemptive relaxation in Section 3 and then develop our solution approach based on Benders decomposition in Section 4. This is followed in Section 5 by an extensive set of computational experiments. We conclude and discuss potential future research directions in Section 6.

**2. Review of Related Literature** Early research on parallel machine scheduling is primarily concerned with the makespan and total (weighted) completion time objectives (Cheng and Sin, 1990). We refer the reader to Pinedo (2008) for a comprehensive discussion of the polynomially solvable cases and structural results of interest for these problems. Some of the more recent and well-known examples of the papers that study $\mathcal{NP}$-complete problems in this domain include van den Akker et al. (1999), Chen and Powell (1999b), Azizoglu and Kirca (1999a), and Azizoglu and Kirca (1999b). Studies on due date related performance measures in the parallel machine environment commenced in earnest in the 1990's and picked up more significantly during the last decade. In this review, we mainly restrict our attention to the literature on parallel machine tardiness and earliness/tardiness scheduling problems with job dependent due dates. This part of the literature creates the context for our study, and we provide a few important pointers otherwise. The great majority of the existing studies on due date related performance measures assumes that the machines are identical, and only a handful of papers consider the case of unrelated parallel machines. For most of the proposed exact approaches, computational scalability remains an issue due to the lack of strong lower bounds. Therefore, we also specifically elaborate on the existing lower bounding methods for parallel machine scheduling problems with additive tardiness and earliness/tardiness objective functions in order to justify our alternate lower bounding scheme introduced in Section 3. See Table 3 in Online Supplement G.3 for a summary of the important points in this section.

The first exact approach for minimizing the total tardiness with distinct due dates on identical parallel machines is due to Azizoglu and Kirca (1998). The authors integrate some dominance rules and a simple bounding technique into a branch-and-bound (B&B) procedure for this problem $Pm//\sum_j T_j$, where $Pm$ in the first field indicates a set of $m$ identical parallel machines. The algorithm is able to handle instances with up to 15 jobs and 3 machines. The lower bound of Azizoglu and Kirca belongs to a very common and simple set of lower bounds which rely on determining a lower bound for the $j$th smallest job completion time $C_{[j]}$, $j = 1,\ldots,n$, among the set of all feasible schedules. These lower bounds on the completion times are then matched with the weights and the due dates in some appropriate order so that the resulting expression yields a lower bound for the problem under consideration. Lower bounding techniques based on such *minimal completion times* are developed or employed in several other papers with tardiness related objectives (Koulamas, 1997, Liaw et al., 2003, Shim and Kim, 2007a,b, Souayah et al., 2009, Yalaoui and Chu, 2002). There is a consensus in the literature that this class of lower bounds is not strong in general. Furthermore, in problems with earliness/tardiness objectives the presence of unforced idle time renders similar lower bounding techniques invalid. For the same problem $Pm//\sum_j T_j$, Yalaoui and Chu (2002) devise another B&B scheme. The limit of this algorithm appears to be 20 jobs and 2 machines within a time limit of 30 minutes. The series of papers by Liaw et al. (2003), Shim and Kim (2007a), and Shim and Kim (2007b) develop a set of closely related optimal methods. Liaw et al. (2003) attack the problem $Rm//\sum_j \pi_j T_j$ of minimizing TWT on unrelated parallel machines. This study appears to be the first exact approach for this problem. The lower bounding scheme is very similar to that in Azizoglu and Kirca (1999b) with a simple enhancement based on the structure of the tardiness objective; however, the method does not scale beyond 4 machines and 18 jobs. Shim and Kim (2007a) tackle the unweighted version $Rm//\sum_j T_j$ in the same machine environment. The proposed B&B method employs some of the existing dominance properties in addition to new ones. The lower bounding technique of Liaw et al. (2003) is adopted, and an alternate lower bound is obtained by reducing the original problem into a

single machine problem by modifying the processing times appropriately and using a previously existing result for the single machine total tardiness problem. The largest problem size that can be handled successfully within 1 hour is 5 machines and 20 jobs. In a similar work, Shim and Kim (2007b) address the problem $Pm//\sum T_j$, and instances with up to 5 machines and 30 jobs are solved optimally within 1 hour. Jouglet and Savourey (2011) devise dominance rules and filtering methods for the problem $Pm/r_j/\sum_j \pi_j T_j$, where the notation $r_j$ in the second field indicates that the release dates may be non-identical, and embed these into a B&B procedure along with an existing lower bound. The authors argue that the lack of good lower bounds prevents them from solving instances with more than 20 jobs and 3 machines. All of the optimal methods discussed so far base their lower bounding efforts on combinatorial arguments that rely on simple properties of the scheduling objectives under consideration. The resulting bounds are generally loose. However, the most promising lower bounds for parallel machine total (weighted) tardiness and earliness/tardiness problems are derived through mathematical programming techniques. For instance, the LP relaxations of the set partitioning formulations of common due date / common due window earliness/tardiness problems solved by column generation yield a prominent class of tight lower bounds (Chen and Lee, 2002, Chen and Powell, 1999a). Bounds obtained from various relaxations of time-indexed formulations are also popular in parallel machine scheduling. An arc-time-indexed formulation whose LP relaxation is tackled by column generation is at the heart of the highly efficient branch-cut-and-price algorithm of Pessoa et al. (2010) for $Pm//\sum \pi_j T_j$. This study is by far the most successful exact algorithm to date on parallel machine tardiness problems and delivers optimal solutions to instances with up to 100 jobs and 4 machines. Tanaka and Araki (2008) apply Lagrangian relaxation to the time-indexed formulation of the problem $Pm//\sum T_j$ in an effort to develop tight lower bounds. Instances with up to 25 jobs and 10 machines are solved optimally. The average gap of the initial lower bound is 2.4% for the instances not solved at the root node. Souayah et al. (2009) take on the weighted version of the problem and study $Pm//\sum_j \pi_j T_j$. With a mix of combinatorial, mathematical programming, and Lagrangian relaxation based lower bounds, about half of the instances with up to 35 jobs and 2 machines are solved to optimality within 20 minutes. We refer the interested reader to the review paper Sen et al. (2003) where the tardiness literature on multi-machine systems is briefly addressed as well. Following this discussion, two observations are due regarding the state of the literature. First, there is a clear need for studying the tardiness related objectives in the unrelated parallel machine environment; we can pinpoint only two studies which focus on the unrelated parallel machine environment. Second, more than 20 to 30 jobs and a few machines seems to be generally beyond the reach for the existing exact methods, attributed to the lack of strong lower bounds. We hope to provide a potential remedy to this issue in this paper.

Several heuristics have been proposed for minimizing the total (weighted) tardiness on identical parallel machines and are reviewed in Online Supplement G.1. For unrelated parallel machines, we are aware of only three papers by Zhou et al. (2007), Mönch (2008), and Lin et al. (2011) which focus on heuristics for $Rm//\sum_j \pi_j T_j$. The first two studies rely on ant colony optimization and benchmark their algorithms against simple heuristics which makes it difficult to evaluate the solution quality in absolute terms. Lin et al. propose a genetic algorithm and two simpler heuristics. The genetic algorithm outperforms all others in the computational experiments and deviates from the optimal solution by 1.8% on average for small instances with 4 machines and 20 jobs. The heuristic that we develop in this paper is scalable to large instances with up to 200 jobs and simultaneously produces both lower and upper bounds of high quality. As evident from the discussion here, this is a significant edge over those in the literature, and we make a valuable contribution to the (unrelated) parallel machine scheduling research with tardiness objectives.

To the best of our knowledge, no exact algorithm has been designed to date for the problem of scheduling a set of independent jobs on a bank of unrelated parallel machines with the objective of minimizing the total (weighted) earliness and tardiness. However, various studies investigate special cases of this problem – see Online Supplement G.2. The most closely related works to our problem *Rm-TWET* are by Kedad-Sidhoum et al. (2008), Mason et al. (2009), and M'Hallah and Al-Khamis (2012). Kedad-Sidhoum et al. experiment with various relaxations of the problem $Pm/r_j/\sum_j \pi_j T_j + \epsilon_j E_j$ by recognizing that the main difficulty in solving earliness/tardiness scheduling problems stems

from the lack of strong lower bounds. The authors extend two classes of lower bounds originally proposed for the single machine case to the identical parallel machine environment. Their discrete assignment-based lower bound is discussed further in Section 3 because it is closely related to our preemptive lower bounding method for *Rm-TWT* and *Rm-TWET*. Kedad-Sidhoum et al. report that the Lagrangian relaxation obtained by dualizing the machine capacity constraints in the time-indexed formulation outperforms others, taking into account both the solution quality and gap. Tanaka and Araki (2008) – discussed previously – employ the same Lagrangian relaxation for $Pm//\sum T_j$. The best bound attained by solving the Lagrangian dual problem in these relaxations is equivalent to that provided by the LP relaxation of the time-indexed formulation. However, solving the Lagrangian dual problem – generally by subgradient optimization – is often computationally more efficient. We also attest to the rapidly increasing computational effort required to solve the LP relaxation of the time-indexed formulation in Section 5. Kedad-Sidhoum et al. obtain upper bounds through a simple local search. Experimental results attest to the quality of both the lower and upper bounds. The average optimality gap attained for instances with up to six machines and 90 jobs is around 1.5%. However, we cite two good reasons for not following a similar path to that of Kedad-Sidhoum et al. and Tanaka and Araki. First, the machine capacity constraints in the time-indexed formulation may be aggregated in the identical parallel machine environment by defining a single resource with a capacity of performing $m$ jobs simultaneously, and this renders the number of dual variables in the Lagrangian relaxation independent from the number of machines in the problem. This, however, is not possible for *Rm-TWT* and *Rm-TWET*, and relaxing the machine capacity constraints – one for each combination of time period and machine – would result in $mH$ dual variables instead of just $H$. Consequently, solving the Lagrangian dual problem would quickly become a formidable task with an increasing number of machines. Second, the solution retrieved from the Lagrangian relaxation does offer little information on how to identify near-optimal job to machine assignments. The job start times provided by the Lagrangian relaxation for a given set of dual multipliers form the basis for a dispatch rule in Tanaka and Araki (2008); however, both these authors and Kedad-Sidhoum et al. need to devise independent heuristics in order to obtain feasible solutions of high-quality for their original problems.

The moving block heuristic of Mason et al. (2009) for $Pm//\sum E_j + T_j$ is tested against an integer programming formulation over instances with up to 40 jobs and 4 machines. The heuristic identifies feasible solutions which are on average better than the incumbent for 20- and 40-job instances. Like Kedad-Sidhoum et al., M'Hallah and Al-Khamis tackle the weighted version of the problem. Their integer programming formulation points out and corrects an error in that of Mason et al. (2009). The limit of the formulation appears to be instances with no more than 20 jobs. In addition, several new heuristics are introduced. The best performing contender turns out to be a hybrid heuristic which is benchmarked against the lower and upper bounds of Kedad-Sidhoum et al. (2008). The hybrid heuristic improves some of the best known solutions for the instances of Kedad-Sidhoum et al.; however, it yields slightly worse solutions on average. The median gap of the hybrid heuristic ranges from 1.4% to 6.1% with respect to the lower bounds of Kedad-Sidhoum et al. (2008) depending on the problem size. It is evident that there is a gap in the literature with respect to the parallel machine earliness/tardiness scheduling problems with distinct due dates. To the best of our knowledge, our work provides the first viable solution approach for the unrelated parallel machine environment in this context.

**3. Problem Statement and Preemptive Relaxation** We consider a bank of $m$ unrelated parallel machines and $n$ jobs, which are all ready at time zero. Each job is processed on exactly one of the machines, where the processing of job $j$ on machine $k$ requires an integer duration of $p_{kj}$ time units. The completion time of job $j$ is denoted by $C_j$. A due date $d_j$ – also assumed to be integral – is associated with each job $j$, and we incur a cost $\pi_j$ per unit time if job $j$ completes processing after $d_j$. Thus, the total weighted tardiness over all jobs is determined as $\sum_j \pi_j T_j$, where the tardiness of job $j$ is calculated as $T_j = \max(0, C_j - d_j)$. For the problem $Rm//\sum_j \pi_j T_j + \epsilon_j E_j$, the objective additionally penalizes the completion of job $j$ prior to its due date $d_j$ at a rate of $\epsilon_j$ per unit time, where the earliness of job $j$ is defined as $E_j = \max(0, d_j - C_j)$. All machines are available continuously from time zero onward, and a machine can execute at most one operation at a time.

An operation must be carried out to completion once started, i.e., preemption is not allowed.

In this section, we introduce our preemptive lower bounding scheme for *Rm-TWT* and *Rm-TWET*. We define two primary design goals for our preemptive relaxation. The tightness of the lower bound is clearly a major concern. Equally important is the information that can be extracted from the optimal solution of the preemptive relaxation to construct feasible solutions of high quality for the original non-preemptive problem. We attain both of these goals – somewhat more successfully for *Rm-TWT* than for *Rm-TWET* from a computational perspective – and demonstrate the effectiveness of the proposed lower and upper bounds in Section 5.

A class of highly efficient lower bounds based on a particular preemption scheme was developed for single machine tardiness and earliness/tardiness scheduling problems during the last decade (Bülbül et al., 2007, Şen and Bülbül, 2012, Sourd and Kedad-Sidhoum, 2003). The key idea of these preemptive relaxations is to divide up jobs with integer processing times into jobs of unit-length and associate a cost with the completion of each of these unit-length jobs. That is, jobs may only be preempted at integer points in time. In this setting, the problem of solving the preemptive relaxation is formulated as an assignment or a transportation problem, where the length of the planning horizon depends on the magnitude of the due dates and the sum of the processing times. Therefore, the formulation size is pseudo-polynomial. On the up side, the availability of very fast algorithms for the assignment and transportation problems does still render this lower bounding technique viable. The formulation (**TR**) below is due to Kedad-Sidhoum et al. (2008), where the original approach in the single machine environment is extended to $m$ identical parallel machines.

$$
\begin{aligned}
(\mathbf{TR}) \qquad & \text{minimize} && \sum_{j=1}^{n}\sum_{t=1}^{H} c'_{jt}x_{jt} && && (1)\\
& \text{subject to} && \sum_{t=1}^{H} x_{jt} = p_j, && j = 1,\ldots,n, && (2)\\
& && \sum_{j=1}^{n} x_{jt} \le m, && t = 1,\ldots,H, && (3)\\
& && 0 \le x_{jt} \le 1, && j = 1,\ldots,n,\ t = 1,\ldots,H. && (4)
\end{aligned}
$$

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

$$
H = \begin{cases} \left\lceil \sum_{j=1}^{n} \max_k \left(p_{jk}\right)/m \right\rceil + p_{\max} & \text{for } Rm\text{-}TWT, \text{ and} \\ \left\lceil \sum_{j=1}^{n} \max_k \left(p_{jk}\right)/m \right\rceil + p_{\max} + d_{\max} & \text{for } Rm\text{-}TWET. \end{cases} \tag{5}
$$

The end of the planning horizon $H$ is determined based on the following observation. For *Rm-TWT* with a regular objective function, all machines are continuously busy until some time $t' \le \left\lceil \sum_{j=1}^{n} \max_k \left(p_{jk}\right)/m \right\rceil$ if at least $m$ jobs are still not completed. Therefore, after time $t'$ the remaining $m-1$ jobs are finished in at most $p_{\max} = \max_{j,k}\left(p_{jk}\right)$ time periods. The end of the planning horizon may thus be set to the value in the first row of (5). An optimal solution of *Rm-TWET*, on the other hand, may include unforced idleness, and the argument just described is only valid if we conservatively assume that all jobs are started at $d_{\max} = \max_j d_j$. Clearly, $p_{\max}$ may be omitted from (5) in the case of a single machine.

If a unit job of job $j$ is executed during the time interval $(t-1, t]$, the decision variable $x_{jt}$ assumes the value one, and the objective is charged a cost of $c'_{jt}$. The constraints (2) mandate that each job $j$ receives $p_j$ units of processing. To observe the machine capacities, constraints (3) require that no more than $m$ unit jobs are processed simultaneously in a given period. Note that the machine index is omitted from the processing times because they are all identical for a given job. Furthermore, no integrality is imposed on the decision variables due to the total unimodularity of the constraint matrix of (**TR**). The optimal objective function value of (**TR**) is a lower bound on that of $Pm//\sum_j \pi_j T_j + \epsilon_j E_j$, as long as the

objective function coefficients satisfy

$$\sum_{s=t-p_j+1}^{t} c'_{js} \leq \epsilon_j (d_j - t)^+ + \pi_j (t - d_j)^+ \quad j = 1, \ldots, n,\ t = p_j, \ldots, H. \tag{6}$$

That is, the total cost incurred in (**TR**) by any job that is scheduled non-preemptively is no larger than that in the original non-preemptive problem (Bülbül et al., 2007). Naturally, the strength of the lower bound depends on the objective coefficients $c'_{jt}$, and this is where the existing works in the literature take different paths. For instance, the cost coefficients of Sourd and Kedad-Sidhoum (2003) satisfy (6) as an equality. Bülbül et al. (2007) characterize and develop an expression for the cost coefficients that are the best among those with a piecewise linear structure with two segments. For these cost coefficients, (6) holds as a strict inequality for some values of $t$. For the one machine problem, these authors also show that the lower bound retrieved from (**TR**) is no better than that provided by the LP relaxation of the time-indexed formulation. Conversely, Pan and Shi (2007) prove the existence of a set of objective coefficients for (**TR**) so that the LP relaxation of the time-indexed formulation and (**TR**) yield identical lower bounds. However, computing the values of these cost coefficients is no less time consuming than solving the LP relaxation of the time-indexed formulation. We also note that the empirical performance of the algorithms based on this set of relaxations is more than satisfactory (Bülbül et al., 2007, Pan and Shi, 2007, Şen and Bülbül, 2012, Sourd and Kedad-Sidhoum, 2003). They strike a good balance between solution quality and time.

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

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

(**TR** − **A**)

$$\text{minimize} \quad \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{t=1}^{H} c'_{jkt} x_{jkt} \tag{7}$$

$$\text{subject to} \quad \sum_{t=1}^{H} x_{jkt} = p_{jk} y_{jk}, \qquad j = 1, \ldots, n,\ k = 1, \ldots, m, \tag{8}$$

$$\sum_{j=1}^{n} x_{jkt} \leq 1, \qquad k = 1, \ldots, m,\ t = 1, \ldots, H, \tag{9}$$

$$\sum_{k=1}^{m} y_{jk} = 1, \qquad j = 1, \ldots, n, \tag{10}$$

$$x_{jkt} \geq 0, \qquad j = 1, \ldots, n,\ k = 1, \ldots, m,\ t = 1, \ldots, H, \tag{11}$$

$$y_{jk} \in \{0, 1\}, \qquad j = 1, \ldots, n,\ k = 1, \ldots, m. \tag{12}$$

(**TR** − **A**) differs from (**TR**) in two main aspects. The capacity constraints (9) appear in a disaggregated form, and all unit jobs of job $j$ are performed on the same machine by constraints (8) and the job partitioning constraints (10). As we hinted at earlier, the cost coefficients $c'_{jkt}$ are of critical importance for the strength of the lower bounds provided by

the preemptive relaxation. In this research, we stick with the cost coefficients by Bülbül et al. (2007) given in (13) and adapted in an obvious way to the unrelated parallel machine environment for two reasons. They empirically outperform those by Sourd and Kedad-Sidhoum (2003) on average (Bülbül et al., 2007, Kedad-Sidhoum et al., 2008), and computing the best set of cost coefficients for a given instance by the method of Pan and Shi (2007) is expensive.

$$c'_{jkt} = \begin{cases} \frac{\epsilon_j}{p_{jk}}\left[(d_j - \frac{p_{jk}}{2}) - (t - \frac{1}{2})\right] & \text{for } t \leq d_j, \text{ and} \\ \frac{\pi_j}{p_{jk}}\left[(t - \frac{1}{2}) - (d_j - \frac{p_{jk}}{2})\right] & \text{for } t > d_j. \end{cases} \tag{13}$$

We next provide a proposition that the optimal solution of (**TR** − **A**) with the cost coefficients given above provides a lower bound on the optimal objective function value of the original problem *Rm-TWT* or *Rm-TWET*. The result is a corollary of Bülbül et al. (2007, Theorem 2.2), where the authors show that the cost coefficients in (13) satisfy (6). The formal proof is in Online Supplement H.1.

Proposition 3.1 *The optimal objective function value of* (**TR** − **A**) *with the cost coefficients given by equation* (13) *is a lower bound on the optimal objective function value of the original non-preemptive problem Rm-TWT or Rm-TWET.*

Our overall strategy for obtaining near-optimal feasible solutions and good lower bounds for *Rm-TWT* and *Rm-TWET* is now clear. We first solve (**TR** − **A**), retrieve the job partition, and then build $m$ individual machine schedules independently. Several heuristics with excellent empirical performance are available for both $1//\sum_j \pi_j T_j$ and $1//\sum_j \pi_j T_j + \epsilon_j E_j$ to perform the latter task. However, in this work we rely on the recent powerful optimal algorithms of Tanaka et al. (2009) and Tanaka and Fujikuma (2012) to handle the single machine problems as we mentioned in Section 1. Our computational experiments in Section 5 ultimately support this decision. Thus, only one major challenge remains. The formulation (**TR** − **A**) is a mixed integer programming problem that is time consuming to solve based on our preliminary computational experiments. However, for a fixed job partition it decomposes into $m$ independent LPs – $m$ independent transportation problems –, and these LPs are solved to optimality very efficiently. These observations suggest that (**TR** − **A**) is amenable to Benders decomposition (Benders, 1962), and developing a Benders decomposition algorithm with *strengthened cuts* for (**TR** − **A**) is our main methodological contribution in this paper.

One final remark is due before we delve into the specifics of our solution method for (**TR** − **A**). For *Rm-TWT*, the formulation may be strengthened by the load balancing constraints (14) which assert that the workloads of two machines cannot differ by more than $p_{\max}$ in an optimal solution of the original non-preemptive parallel machine scheduling problem. Otherwise, we could transfer the final job on one of these machines to the other one without degrading the objective function value. Note that similar concepts have been incorporated into various properties and dominance rules elsewhere in the literature (Azizoglu and Kirca, 1999b, Theorem 1). However, for *Rm-TWET* with a non-regular objective function, we can easily create instances for which no optimal solution satisfies (14).

$$-p_{\max} \leq \sum_{j=1}^{n} p_{jk} y_{jk} - \sum_{j=1}^{n} p_{jl} y_{jl} \leq p_{\max}, \qquad k = 1, \ldots, m-1,\ l = k+1, \ldots, m. \tag{14}$$

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

**4. Benders Decomposition** Parallel machine scheduling problems have a partitioning and a scheduling component. That is, if we assign jobs to machines by fixing the variables $y_{jk}, j = 1, \ldots, n, k = 1, \ldots, m$, so that the constraints (10) are satisfied, then the model (**TR** − **A**) decomposes into $m$ independent transportation problems. We exploit this key observation to design an algorithm based on Benders decomposition for solving (**TR** − **A**) efficiently. To this end, we reformulate (**TR** − **A**) for a fixed $\overline{y}$ by replacing the right hand side of the set of constraints (8) by $p_{jk}\overline{y}_{jk}$ and dropping the set of constraints (10) and (12) from the model. In the resulting linear program (**TR** − **A**($\overline{y}$)), $u_{jk},\ j = 1, \ldots, n,\ k = 1, \ldots, m$, and $v_{kt},\ k = 1, \ldots, m,\ t = 1, \ldots, H$, are the dual variables associated with the set of constraints (8) and (9), respectively.

The dual of (**TR** − **A**($\overline{y}$)) is then stated below, where the decomposition into $m$ independent transportation problems is made explicit:

$$z(\overline{y}) = \sum_{k=1}^{m} z_k(\overline{y}), \tag{15}$$

where

$$(\mathbf{DS}_k) \qquad z_k(\overline{y}) = \text{maximize} \quad \sum_{j=1}^{n} p_{jk}\overline{y}_{jk}u_{jk} + \sum_{t=1}^{H} v_{kt} \tag{16}$$

$$\text{subject to} \quad u_{jk} + v_{kt} \leq c'_{jkt}, \qquad j = 1,\ldots,n,\ t = 1,\ldots,H, \tag{17}$$

$$v_{kt} \leq 0, \qquad t = 1,\ldots,H, \tag{18}$$

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

Based on the objective function (16) of (**DS**$_k$), we obtain the following restricted Benders master problem (**RMP**), where $C$ denotes the current number of times the cut generation subproblems (**TR**$_k$), $k = 1,\ldots,m$, have been solved. The optimal values of the dual variables $\overline{u}_{jk}$, $j = 1,\ldots,n, k = 1,\ldots,m$, and $\overline{v}_{kt}, k = 1,\ldots,m, t = 1,\ldots,H$, in round $c$ of the cut generation are represented by $\overline{u}^c_{jk}$ and $\overline{v}^c_{kt}$, respectively. The auxiliary variable $\eta_k$ indicates a lower bound on the total cost incurred by the jobs assigned to machine $k$, and the objective function value $\sum_{k=1}^{m} \eta_k$ of (**RMP**) is therefore a lower bound on the optimal objective values of (**TR** − **A**) and the original non-preemptive scheduling problem *Rm-TWT* or *Rm-TWET*.

$$(\mathbf{RMP}) \qquad \text{minimize} \quad \sum_{k=1}^{m} \eta_k \tag{19}$$

$$\text{subject to} \quad \sum_{k=1}^{m} y_{jk} = 1, \qquad j = 1,\ldots,n, \tag{20}$$

$$\eta_k \geq \sum_{j=1}^{n} p_{jk}\overline{u}^c_{jk}y_{jk} + \sum_{t=1}^{H} \overline{v}^c_{kt}, \qquad k = 1,\ldots,m,\ c = 1,\ldots,C, \tag{21}$$

$$y_{jk} \in \{0,1\}, \qquad j = 1,\ldots,n,\ k = 1,\ldots,m. \tag{22}$$

Note that (**TR** − **A**($\overline{y}$)) is feasible and (**DS**$_k$), $k = 1,\ldots,m$, is bounded for any $\overline{y}$ that satisfies the constraints (10). Therefore, no feasibility cuts are required, and only optimality cuts are generated and added iteratively to (**RMP**) during the course of the algorithm. Furthermore, the cut generation subproblem (**TR**$_k$) for machine $k$ includes all jobs and is solved by considering the full length of the planning horizon. From a computational point of view, however, we are better off by defining the set of jobs to be processed on machine $k$ as $J_k = \{j \mid y_{jk} = 1\}$, setting the last period of processing on machine $k$ – designated by $H_k$ – as appropriate based on (5), and then solving a restricted version of (**TR**$_k$) over these jobs and time periods only. This restricted cut generation subproblem formulation and the corresponding dual slave problem are referred to as (**TR**$_k$ − **R**) and (**DS**$_k$ − **R**), respectively. Obviously, the optimal solution of (**TR**$_k$ − **R**) may be extended to an optimal solution of (**TR**$_k$) trivially by setting $x_{jkt} = 0$ for $j \in J_k,\ t = H_k + 1,\ldots,H$, and $j \notin J_k,\ t = 1,\ldots,H$. The relationship between the optimal solutions of (**DS**$_k$) and (**DS**$_k$ − **R**) and its implications for the dynamic generation of the constraints (21) require a deeper discussion which is relegated to the next section.

In the *multi-cut* restricted master problem formulation (**RMP**) above, we approximate the objective function of (**TR** − **A**) by estimating the cost accumulated on each machine separately as evident from the set of constraints (21). Alternatively, we could have employed a weaker *single-cut* version of the restricted master problem by aggregating all $m$ cuts generated after solving (**TR**$_k$), $k = 1,\ldots,m$, and replacing $\sum_{k=1}^{m} \eta_k$ by a single variable $\eta$ in the formulation as appropriate. The single-cut version results in fast solution times for the restricted master problem at the expense of more iterations overall. Ultimately, the trade-off between these two alternatives is only decided during the computations. In our preliminary

testing, the cut generation algorithm based on (**RMP**) was clearly superior to that based on the single-cut version in terms of speed. Thus, the rest of the paper is focused exclusively on (**RMP**). The pseudo-code of the cut generation procedure is stated in Algorithm 1 in Online Supplement I.

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

$$\eta_k \geq \sum_{j \in J_k} p_{jk}\overline{u}_{jk}y_{jk} + \sum_{t=1}^{H_k} \overline{v}_{kt} \tag{23}$$

produced directly out of an optimal solution of ($\mathbf{DS}_k - \mathbf{R}$) relies on the assumption that augmenting this solution trivially with $\overline{u}_{jk} = 0$ for $j \notin J_k$ and $\overline{v}_{kt} = 0$ for $t = H_k + 1, \ldots, H$, is feasible with respect to ($\mathbf{DS}_k$). It is a simple matter to show that as long as the optimal solution of ($\mathbf{DS}_k - \mathbf{R}$) satisfies $\max_{t=1,\ldots,H_k} \overline{v}_{kt} = 0$ (see Lemma 4.1), this augmented solution is feasible with respect to ($\mathbf{DS}_k$) if we are solving an instance of *Rm-TWT* because the cost coefficients $c'_{jkt}$ are non-negative and non-decreasing over time. However, the trivial augmentation is not necessarily feasible for every instance of *Rm-TWET*, and (23) might therefore be an invalid Benders cut. To illustrate, consider an instance of *Rm-TWET* and assume that for some assignment $\overline{\boldsymbol{y}}$ of the jobs to the machines we solve ($\mathbf{TR}_k - \mathbf{R}$) with $H_k \leq d_j - 1$, where $j \notin J_k$ and $p_{jk} > 1$. In the trivially augmented solution for ($\mathbf{DS}_k$), constraint (17) for job $j$ and time period $d_j$ is violated because $c'_{jkd_j} = \frac{\epsilon_j}{p_{jk}}\left(\frac{1}{2} - \frac{p_{jk}}{2}\right) < 0$ for $\epsilon_j > 0$ and $p_{jk} > 1$, and $\overline{u}_{jk} + \overline{v}_{kd_j} = 0 + 0 \leq c'_{jkd_j}$ does not hold. Therefore, we need a mechanism which can always extend an optimal solution of ($\mathbf{DS}_k - \mathbf{R}$) to an optimal solution of ($\mathbf{DS}_k$). Proposition 4.1 proves that the cut strengthening procedure described next fulfills this goal. This ensures that the dual slave problem is always solved over the same feasible region and the generated Benders cuts are valid.

Several papers in the literature report that a straightforward implementation of Benders decomposition yields a dismal performance from a computational point of view (Fischetti et al., 2010, Magnanti and Wong, 1981, Üster and Agrahari, 2011, Van Roy, 1986, Wentges, 1996). This is often rooted in the primal degeneracy in the cut generation subproblem which implies the existence of multiple optimal solutions to the dual slave problem. That is, possibly several alternate cuts may be generated based on the same master problem solution, and the particular choice has a profound impact on the computational performance. These concerns are also valid for us because the transportation problem suffers from a well-known primal degeneracy. To address these issues, we initially adapted the generic Benders cut strengthening method introduced recently by Fischetti et al. (2010) to our problem. These authors argue that identifying a small set of constraints in the subproblem that allows us to cut the current master solution is of practical interest to enhance the computational performance. To this end, they pose the cut generation subproblem as a pure feasibility problem and look for a *minimal infeasible subsystem* of small cardinality. However, applying this technique to our problem does not preserve the transportation problem structure in the cut generation subproblems. This results in substantially prolonged subproblem solution times with ultimately uncompetitive overall performance for Benders decomposition. Instead, here we follow an approach that is similar to those of Üster and Agrahari (2011), Van Roy (1986) to strengthen our Benders cuts, which also resolves the issue pointed out in the previous paragraph regarding the validity of the cuts constructed based on an optimal solution of ($\mathbf{DS}_k - \mathbf{R}$). We reap great savings in solution time from this enhancement. In fact, our

algorithm exhibits very poor convergence without this cut strengthening.

The key to showing the validity of our cut generation as well as strengthening the Benders cuts is to prove that we can always augment an optimal solution of $(\mathbf{DS_k} - \mathbf{R})$ to obtain a feasible solution of $(\mathbf{DS_k})$ with the same objective function value. This would establish that the augmented solution is optimal for $(\mathbf{DS_k})$ because $\overline{y}_{jk} = 0$ for all $j \notin J_k$ and $v_{kt} \leq 0$ for all $t = H_k + 1, \ldots, H$ (see Proposition 4.1). Compared to (21), the benefit is that we can produce a strengthened Benders cut of the form

$$\eta_k \geq \sum_{j=1}^{n} p_{jk} \overline{u}''_{jk} y_{jk} + \sum_{t=1}^{H} \overline{v}''_{kt} \tag{24}$$

from an optimal solution $(\overline{u}''_k, \overline{v}''_k)$ of $(\mathbf{DS_k})$ so that $\overline{u}''_{jk} \neq 0$ for $j \notin J_k$ in general. We first need the following result to attain our goal. The formal proof is in Online Supplement H.2.

Lemma 4.1 *There exists an optimal solution* $(\overline{u}'_k, \overline{v}'_k)$ *to* $(\mathbf{DS_k} - \mathbf{R})$ *such that* $\max_{t=1,\ldots,H_k} \overline{v}'_{kt} = 0$.

Assume that we are given an optimal solution $(\overline{u}'_k, \overline{v}'_k)$ of $(\mathbf{DS_k} - \mathbf{R})$ which satisfies the property in Lemma 4.1 and a corresponding Benders cut of the form (23). Clearly, we can always extend the planning horizon in $(\mathbf{DS_k} - \mathbf{R})$ to $1, \ldots, H$, and augment this optimal solution with zeros as necessary and still preserve the optimality. Therefore, without loss of generality assume that an augmented optimal solution $(\overline{u}'_k, \overline{v}''_k)$ is available to $(\mathbf{DS_k} - \mathbf{R})$, where $\overline{v}''_{kt} = \overline{v}'_{kt}$ for $t = 1, \ldots, H_k$, and $\overline{v}''_{kt} = 0$ for $t = H_k + 1, \ldots, H$. Based on this augmented optimal solution, we next explain how an original Benders cut of the form (23) is strengthened, and then prove that this strengthened cut corresponds to an optimal solution of $(\mathbf{DS_k})$ and is therefore valid.

The variables $u_{jk}$, $j \notin J_k$, do not appear in $(\mathbf{DS_k} - \mathbf{R})$ and are implicitly assumed to be zero. Consequently, no term appears on the right hand side of a Benders cut (23) for the jobs that are assigned to other machines in the current restricted master solution $\overline{y}$. However, $\overline{y}_{jk} = 0$ for all such jobs $j \notin J_k$, and we can produce a stronger cut by incorporating $y_{jk}$, $j \notin J_k$, into the right hand side of (23) with positive coefficients $p_{jk}\overline{u}''_{jk}$, $j \notin J_k$, if possible. In order to compute a good set of values $\overline{u}''_{jk}$, $j \notin J_k$, we solve the following optimization problem for a given augmented optimal solution $(\overline{u}'_k, \overline{v}''_k)$ of $(\mathbf{DS_k} - \mathbf{R})$:

$$\text{maximize} \quad \sum_{j \notin J_k} p_{jk} u_{jk} \tag{25}$$

$$\text{subject to} \quad u_{jk} \leq c'_{jkt} - \overline{v}''_{kt}, \qquad j \notin J_k,\ t = 1, \ldots, H. \tag{26}$$

The constraints (26) are required to establish that the coefficients of the strengthened cut correspond to an optimal solution of $(\mathbf{DS_k})$ – see Proposition 4.1. Clearly, (25)-(26) decomposes by job, and the optimal solution is determined as:

$$\overline{u}''_{jk} = \min \left\{ \min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}), \min_{t=H_k+1,\ldots,H} c'_{jkt} \right\}, \qquad j \notin J_k. \tag{27}$$

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

$$\min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}) \leq \max_{t=1,\ldots,H_k} c'_{jkt} \leq \min_{t=H_k+1,\ldots,H} c'_{jkt}. \tag{28}$$

Consequently, (27) simplifies to

$$\overline{u}''_{jk} = \min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}), \qquad j \notin J_k, \tag{29}$$

for *Rm-TWT*.

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

$$\overline{u}''_{jk} = \left\{ \begin{array}{ll} \min\left( \min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}), c'_{jkH_{k+1}} \right) & \text{if } H_k \geq d_j \\ \min\left( \min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}), c'_{jkd_j} \right) & \text{if } H_k \leq d_j - 1 \end{array} \right\}, \qquad j \notin J_k. \tag{30}$$

Thus, the strengthened cut finally takes the form specified in (24), where $\overline{u}''_{jk} = \overline{u}'_{jk}$ for $j \in J_k$ and $\overline{u}''_{jk}$, $j \notin J_k$, is calculated based on either (29) or (30), respectively, depending on whether we solve an instance of *Rm-TWT* or *Rm-TWET*. We next prove that this augmented solution $(\overline{\boldsymbol{u}}''_k, \overline{\boldsymbol{v}}''_k)$ is optimal for $(\mathbf{DS}_k)$.

PROPOSITION 4.1 *The dual variables* $(\overline{\boldsymbol{u}}''_k, \overline{\boldsymbol{v}}''_k)$*, which produce a strengthened Benders cut* (24)*, are optimal with respect to* $(\mathbf{DS}_k)$.

PROOF. Recall that $(\overline{\boldsymbol{u}}''_k, \overline{\boldsymbol{v}}''_k)$ is constructed by augmenting an optimal solution $(\overline{\boldsymbol{u}}'_k, \overline{\boldsymbol{v}}'_k)$ of $(\mathbf{DS_k - R})$ which satisfies the property in Lemma 4.1. Therefore, $\overline{u}''_{jk} + \overline{v}''_{kt} \leq c'_{jkt}$, $j \in J_k, t = 1, \ldots, H_k$, and $\overline{v}''_{kt} \leq 0$, $t = 1, \ldots, H_k$, hold automatically. In addition, $\overline{v}''_{kt}$, $t = H_k + 1, \ldots, H$, are set directly to zero. Therefore, we only need to verify that $\overline{u}''_{jk} + \overline{v}''_{kt} \leq c'_{jkt}$, $j \in J_k, t = H_k + 1, \ldots, H$, and $\overline{u}''_{jk} + \overline{v}''_{kt} \leq c'_{jkt}$, $j \notin J_k, t = 1, \ldots, H$, to show the feasibility of $(\overline{\boldsymbol{u}}''_k, \overline{\boldsymbol{v}}''_k)$ for $(\mathbf{DS}_k)$. The latter inequalities are enforced directly by the constraints (26). For the former, note that for any job $j \in J_k$ the end of the planning horizon $H_k$ is larger than $d_j$ in both *Rm-TWT* and *Rm-TWET*. Then, by a similar argument that leads to (28), $\overline{u}''_{jk} \leq \max_{t'=1,\ldots,H_k} c'_{jkt'}$, and we obtain $\overline{u}''_{jk} + \overline{v}''_{kt} = \overline{u}''_{jk} \leq \max_{t'=1,\ldots,H_k} c'_{jkt'} \leq c'_{jkt}$ for all time periods $t = H_k + 1, \ldots, H$, as desired.

The optimal objective function value of $(\mathbf{DS}_k)$ is bounded from above by that of $(\mathbf{DS_k - R})$ because all constraints of $(\mathbf{DS_k - R})$ are present in (17)-(18), $\overline{y}_{jk} = 0$ for $j \notin J_k$, and $\sum_{t=H_k+1}^{H} v_{kt} \leq 0$. This completes the proof since the objective function value associated with $(\overline{\boldsymbol{u}}''_k, \overline{\boldsymbol{v}}''_k)$ in $(\mathbf{DS}_k)$ is clearly identical to that associated with the optimal solution $(\overline{\boldsymbol{u}}'_k, \overline{\boldsymbol{v}}'_k)$ in $(\mathbf{DS_k - R})$. □

The pseudo-code of our Benders decomposition scheme with the cut strengthening feature for solving $(\mathbf{TR - A})$ is stated in Algorithms 1-2 in Online Supplement I, and Proposition 4.1 proves its correctness. The cut strengthening specified by the Steps 3-6 in Algorithm 2 has a pseudo-polynomial time complexity of $O(nH)$ with an overall complexity of $O(mnH)$ for $m$ machines. In practice, it is very fast.

In classical textbook applications of Benders decomposition, the current restricted master problem is solved to optimality and then cuts generated based on this optimal solution are added to it before the restricted master problem is re-optimized. This loop is repeated until the optimality gap of **(RMP)** – the expression $\frac{z(\overline{\boldsymbol{y}}) - \sum_{k=1}^{m} \overline{\eta}_k}{\sum_{k=1}^{m} \overline{\eta}_k}$ – is smaller than a prespecified tolerance level, where the current optimal objective $\sum_{k=1}^{m} \overline{\eta}_k$ of **(RMP)** is a lower bound on that of $(\mathbf{TR - A})$ and $z(\overline{\boldsymbol{y}})$ is the objective value of a feasible solution of $(\mathbf{TR - A})$. The primary drawback of this classical scheme is that a new search tree is constructed every time the restricted master problem is solved (Rubin, 2011). Consequently, valuable time may be expended toward re-evaluating the same nodes over and over again. In contrast, using the *lazy constraint* technology offered by the state-of-the-art solvers allows us to execute the entire algorithm on a single search tree (IBM ILOG CPLEX, 2011). In Step 11 of Algorithm 1, we invoke the *lazy constraint callback* routine for every candidate incumbent solution. The callback routine either identifies a missing Benders cut violated by the candidate solution and introduces it as a lazy constraint into the model or certifies the candidate as valid. Ultimately, no integer solution is evaluated multiple times during the course of the algorithm. Moreover, labeling the generated cuts as lazy informs the solver that most of such constraints are not expected to be active at the optimal solution. Thus, we fully exploit the capabilities of the solver and allow it to apply the generated cuts as it deems necessary. The use of the lazy constraint technology appears to be relatively rare in the operations research literature, and we hope that it will be employed more frequently in the future given that it may unleash the power of a cut generation algorithm which seems impractical otherwise.

**5. Computational Results** Outstanding among the accomplishments of this research is that both *Rm-TWT* with a regular scheduling objective (see Section 5.1) and *Rm-TWET* with a non-regular scheduling objective (see Online Supplement J.2) are tackled successfully by the exact same solution approach. For both problems, the overarching goal of our computational study is to demonstrate that the proposed Benders-type method solves the preemptive relaxation $(\mathbf{TR - A})$ to (near-) optimality in short computation times and provides tight lower bounds as well as high quality job partitions for the original problems. Very large instances of both problems are within the reach of our algorithm; however,

we concede that the performance is somewhat better for *Rm-TWT* than for *Rm-TWET*.

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

**Table 1** Instance generation parameters.

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

There are 12 combinations of the TF, RDD values and for each combination, 5 instances are generated. Therefore, we create 60 instances for each pair of $m$, $n'$ values and a total of 720 instance pairs. The instances in a pair are identical, except that $\epsilon_j = 0$, $j = 1,\dots,n$, in the *Rm-TWT* instance. This data generation scheme allows us to draw clear conclusions about the relative difficulty of *Rm-TWET* with respect to *Rm-TWT*. As pointed out by Kedad-Sidhoum et al. (2008), the motivation for the relatively large TF and small RDD values is that in most practical production environments the due dates are not loose and not distant from each other. The rationale behind the selected $[\alpha,\beta]$ values reflects that the earliness cost is typically regarded as a finished goods inventory holding cost and should be less than the cost of loss of customer goodwill or a contractual penalty represented by the tardiness cost.

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

To justify the use of the proposed Benders-type approach to solve (**TR** − **A**), we benchmark it against (**TR** − **A**) - **CPX**, where the monolithic formulation (**TR** − **A**) is solved directly by invoking `CPLEX`. In this case, we let `CPLEX` decide whether to apply its dynamic search by running it with the default parameter settings, except that the opportunistic parallel search mode is turned on for a head-to-head comparison with (**TR** − **A**) - **BDS**. The relative gap tolerance parameter `EpGap` of

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

All formulations are solved within the same working memory limit of 15 GB (`WorkMem`=15,000). However, the memory footprint of (**TR − A**) - **BDS** does not exceed a few gigabytes even for the largest instances with 200 jobs and 5 machines. The maximum number of threads that CPLEX is allowed to use – governed by the parameter `Threads` – is seven for all methods. The time limit parameter `TiLim` takes on the values 1800, 1800 and 600 seconds for (**TI**), (**TR − A**) - **CPX**, and (**TR − A**) - **BDS**, respectively.

The next section reports the results obtained for *Rm-TWT*, and the results for *Rm-TWET* are relegated to Online Supplement J.2. For ease of perusal, all tables employ a color formatting scheme so that the values of a performance indicator ranging from better to worse are indicated with colors changing from green towards red.

**5.1 Results for *Rm-TWT*** Table 2 consists of 12 parts, one for each possible combination of $n$ and $m$ listed in the first two columns. We report three types of percentage gaps in the table, labeled as "(**TR − A**) - **BDS**", "LB Quality", and "Feasible Sol'n" in Columns 4–12. The average times needed to solve the preemptive relaxation (**TR − A**) to within 3% of optimality by (**TR − A**) - **CPX** and (**TR − A**) - **BDS** are presented in Columns 13–18. The color formatting is applied to these two sets of columns together to facilitate a head-to-head comparison. For each performance indicator, detailed results for each possible combination of TF and RDD values are included. The TF values appear in the third column, and the RDD values are specified in the column headers. All gaps larger than 100% are set to 100%, and the gap of a feasible solution with a positive objective function value with respect to a lower bound of zero is assumed to be 100%. Each value in the table represents an average over five instances based on our data generation scheme discussed previously.

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

The next three columns under "LB Quality" attest to the quality of the lower bound (LB) provided by (**TR − A**) - **BDS** for the optimal objective value of *Rm-TWT*. For a given instance, the expression $\frac{(\texttt{``Best Integer''}-\texttt{``LB''})}{(\texttt{``Best Bound''})}$ provides an upper bound on the gap of LB, where `''Best Integer''` and `''Best Bound''` are the objective function values associated with the best feasible solution available – retrieved from either our approach or (**TI**) – and the best lower bound provided by any one of the methods (**TR − A**) - **CPX**, (**TR − A**) - **BDS**, or (**TI**), respectively. For any $n$, $m$ combination, the average LB gap summarized across all TF and RDD values does not exceed 8.15%, and the average LB gap across all instances is just 5.64%. In fact, only 8% of the instances (58 instances) have an LB gap larger than 10%.

The following three columns under "Feasible Sol'n" present the average upper bounds on the optimality gaps attained

**Table 2** Results for *Rm-TWT*.

| | | RDD | Percentage Gaps | | | | | | | | | (**TR** − **A**) - Solution Times | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | (**TR** − **A**) - **BDS** | | | LB Quality | | | Feasible Sol'n | | | **CPX** | | | **BDS** | | |
| $n$ | $m$ | TF | 0.2 | 0.4 | 0.6 | 0.2 | 0.4 | 0.6 | 0.2 | 0.4 | 0.6 | 0.2 | 0.4 | 0.6 | 0.2 | 0.4 | 0.6 |
| 40 | 2 | 0.4 | 1.3 | 1.6 | 0.5 | 3.0 | 7.5 | 13.2 | 1.6 | 4.5 | 6.3 | 2 | 3 | 10 | 2 | 2 | 3 |
| | | 0.6 | 2.0 | 2.2 | 2.2 | 2.8 | 3.5 | 6.4 | 1.0 | 1.8 | 2.2 | 4 | 3 | 4 | 2 | 2 | 3 |
| | | 0.8 | 2.5 | 2.6 | 2.4 | 2.8 | 3.6 | 3.7 | 0.8 | 0.6 | 0.6 | 6 | 5 | 5 | 1 | 1 | 3 |
| | | 1.0 | 2.4 | 2.1 | 2.2 | 2.3 | 2.4 | 2.3 | 0.5 | 0.5 | 0.8 | 6 | 6 | 5 | 1 | 1 | 1 |
| 60 | 2 | 0.4 | 2.2 | 1.8 | 1.6 | 3.5 | 5.3 | 9.5 | 0.9 | 2.0 | 5.0 | 8 | 7 | 11 | 5 | 7 | 10 |
| | | 0.6 | 2.4 | 2.4 | 2.6 | 3.3 | 4.4 | 5.7 | 0.9 | 1.3 | 1.7 | 15 | 12 | 11 | 5 | 7 | 18 |
| | | 0.8 | 2.7 | 1.7 | 2.5 | 3.1 | 2.7 | 4.4 | 1.2 | 0.6 | 1.2 | 21 | 19 | 17 | 3 | 6 | 9 |
| | | 1.0 | 2.2 | 2.8 | 2.8 | 1.9 | 2.2 | 2.5 | 0.6 | 1.2 | 1.0 | 24 | 25 | 24 | 2 | 2 | 2 |
| | 3 | 0.4 | 2.4 | 2.1 | 1.4 | 5.0 | 9.3 | 42.9 | 1.9 | 3.6 | 36.8 | 20 | 60 | 215 | 2 | 4 | 4 |
| | | 0.6 | 2.7 | 2.7 | 2.8 | 3.8 | 5.0 | 8.7 | 1.5 | 1.7 | 3.0 | 32 | 30 | 36 | 1 | 2 | 12 |
| | | 0.8 | 2.8 | 2.8 | 2.6 | 3.1 | 4.3 | 4.8 | 1.4 | 0.9 | 0.9 | 40 | 40 | 36 | 1 | 2 | 3 |
| | | 1.0 | 2.6 | 2.5 | 2.5 | 2.2 | 2.6 | 3.2 | 0.8 | 0.6 | 0.8 | 41 | 44 | 41 | 1 | 1 | 1 |
| 80 | 2 | 0.4 | 2.0 | 2.2 | 2.0 | 3.0 | 5.0 | 11.6 | 1.1 | 2.0 | 6.8 | 15 | 12 | 44 | 8 | 17 | 20 |
| | | 0.6 | 2.2 | 1.4 | 2.6 | 2.9 | 2.8 | 4.2 | 1.0 | 1.1 | 0.9 | 33 | 27 | 30 | 10 | 21 | 36 |
| | | 0.8 | 2.9 | 2.3 | 2.7 | 3.3 | 3.0 | 4.3 | 1.5 | 0.7 | 1.1 | 52 | 56 | 44 | 3 | 9 | 10 |
| | | 1.0 | 2.2 | 2.4 | 2.3 | 1.6 | 2.7 | 2.7 | 1.1 | 0.4 | 0.6 | 69 | 63 | 59 | 3 | 6 | 7 |
| | 4 | 0.4 | 2.7 | 2.6 | 1.2 | 6.8 | 10.4 | 38.0 | 3.7 | 5.0 | 36.6 | 42 | 295 | 1716 | 5 | 7 | 11 |
| | | 0.6 | 2.7 | 2.5 | 3.8 | 4.2 | 5.5 | 10.1 | 1.6 | 2.1 | 4.2 | 79 | 73 | 484 | 2 | 7 | 462 |
| | | 0.8 | 2.8 | 2.4 | 2.9 | 3.6 | 4.3 | 5.9 | 1.1 | 0.7 | 1.0 | 111 | 104 | 98 | 1 | 6 | 11 |
| | | 1.0 | 2.5 | 2.8 | 2.8 | 2.2 | 3.1 | 3.7 | 0.7 | 1.1 | 0.7 | 118 | 112 | 121 | 1 | 1 | 2 |
| 90 | 3 | 0.4 | 2.2 | 2.2 | 0.9 | 3.8 | 7.3 | 26.9 | 1.6 | 3.5 | 21.2 | 34 | 112 | 1030 | 5 | 11 | 28 |
| | | 0.6 | 2.6 | 2.8 | 2.9 | 3.3 | 4.3 | 6.3 | 1.2 | 1.4 | 2.1 | 93 | 87 | 93 | 2 | 5 | 24 |
| | | 0.8 | 2.6 | 2.5 | 2.6 | 3.3 | 3.8 | 4.5 | 1.1 | 0.9 | 1.2 | 122 | 121 | 117 | 2 | 3 | 5 |
| | | 1.0 | 2.6 | 2.4 | 2.4 | 2.1 | 3.0 | 3.2 | 1.2 | 0.8 | 1.0 | 148 | 142 | 138 | 2 | 2 | 2 |
| 100 | 5 | 0.4 | 2.7 | 2.7 | 2.8 | 7.3 | 13.7 | 20.0 | 6.2 | 13.3 | 20.0 | 104 | 1119 | 1800 | 6 | 13 | 125 |
| | | 0.6 | 2.7 | 2.8 | 5.9 | 4.2 | 5.7 | 12.8 | 2.9 | 4.6 | 9.3 | 149 | 168 | 883 | 4 | 15 | 601 |
| | | 0.8 | 2.6 | 2.8 | 3.1 | 3.7 | 4.9 | 6.5 | 2.3 | 3.5 | 4.9 | 235 | 182 | 242 | 2 | 15 | 308 |
| | | 1.0 | 2.5 | 2.5 | 2.8 | 2.8 | 3.3 | 4.3 | 1.4 | 1.9 | 2.7 | 253 | 233 | 235 | 2 | 2 | 3 |
| 120 | 3 | 0.4 | 2.5 | 1.9 | 1.1 | 3.7 | 6.5 | 42.5 | 2.3 | 5.8 | 42.5 | 48 | 66 | 1367 | 7 | 20 | 109 |
| | | 0.6 | 2.2 | 2.6 | 2.4 | 2.9 | 4.2 | 5.6 | 1.6 | 2.9 | 4.6 | 205 | 172 | 162 | 4 | 8 | 49 |
| | | 0.8 | 2.6 | 2.9 | 2.7 | 3.0 | 4.0 | 4.5 | 1.5 | 2.3 | 2.8 | 262 | 254 | 221 | 4 | 5 | 8 |
| | | 1.0 | 2.8 | 2.3 | 2.6 | 3.0 | 2.7 | 3.3 | 1.1 | 1.3 | 1.4 | 338 | 344 | 303 | 3 | 4 | 6 |
| | 4 | 0.4 | 2.2 | 2.9 | 10.8 | 4.1 | 9.1 | 20.0 | 3.0 | 7.9 | 20.0 | 96 | 516 | 1816 | 13 | 18 | 135 |
| | | 0.6 | 2.7 | 2.6 | 3.1 | 3.6 | 4.4 | 7.4 | 2.4 | 2.8 | 5.9 | 205 | 209 | 327 | 4 | 18 | 225 |
| | | 0.8 | 2.5 | 2.5 | 2.9 | 3.2 | 4.1 | 5.0 | 1.9 | 2.6 | 3.2 | 316 | 315 | 287 | 3 | 6 | 16 |
| | | 1.0 | 2.7 | 2.7 | 2.7 | 3.0 | 3.2 | 3.7 | 1.4 | 1.6 | 2.3 | 346 | 326 | 291 | 2 | 3 | 3 |
| 150 | 5 | 0.4 | 2.5 | 2.9 | 0.0 | 5.2 | 10.0 | 0.0 | 4.0 | 9.0 | 0.0 | 216 | 1356 | 1527 | 19 | 35 | 18 |
| | | 0.6 | 2.7 | 2.7 | 3.8 | 3.7 | 4.6 | 8.8 | 2.1 | 3.3 | 6.6 | 380 | 379 | 804 | 5 | 39 | 415 |
| | | 0.8 | 2.6 | 2.7 | 2.9 | 3.3 | 4.3 | 5.5 | 1.8 | 2.8 | 4.1 | 683 | 646 | 600 | 4 | 11 | 48 |
| | | 1.0 | 1.3 | 2.5 | 2.4 | 1.6 | 3.1 | 3.5 | 0.8 | 1.7 | 2.3 | 752 | 713 | 653 | 5 | 4 | 4 |

Continued on next page. . .

Table 2 continued...

| $n$ | $m$ | RDD \ TF | Percentage Gaps: (**TR − A**) - **BDS** 0.2 | 0.4 | 0.6 | LB Quality 0.2 | 0.4 | 0.6 | Feasible Sol'n 0.2 | 0.4 | 0.6 | (**TR − A**) - Solution Times: **CPX** 0.2 | 0.4 | 0.6 | **BDS** 0.2 | 0.4 | 0.6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 160 | 4 | 0.4 | 2.8 | 2.8 | 0.0 | 4.5 | 9.2 | 0.0 | 3.1 | 7.9 | 0.0 | 143 | 451 | 1361 | 9 | 40 | 29 |
| | | 0.6 | 2.7 | 2.9 | 2.9 | 3.3 | 4.2 | 6.4 | 1.6 | 2.7 | 4.8 | 429 | 302 | 704 | 6 | 11 | 176 |
| | | 0.8 | 2.6 | 2.2 | 2.6 | 3.1 | 3.3 | 4.3 | 1.6 | 2.2 | 2.8 | 713 | 742 | 668 | 5 | 9 | 12 |
| | | 1.0 | 2.0 | 2.0 | 2.6 | 2.2 | 2.4 | 3.3 | 0.9 | 1.1 | 1.8 | 934 | 836 | 851 | 5 | 6 | 6 |
| 200 | 5 | 0.4 | 2.5 | 3.4 | 0.0 | 4.5 | 12.3 | 0.0 | 3.3 | 10.8 | 0.0 | 345 | 1476 | 793 | 24 | 324 | 35 |
| | | 0.6 | 2.5 | 2.8 | 3.9 | 3.2 | 4.5 | 7.9 | 1.7 | 2.7 | 5.4 | 964 | 752 | 1327 | 11 | 33 | 510 |
| | | 0.8 | 2.6 | 2.7 | 2.7 | 3.1 | 3.9 | 4.6 | 1.9 | 2.2 | 3.0 | 1720 | 1708 | 1440 | 7 | 29 | 36 |
| | | 1.0 | 2.4 | 2.0 | 2.3 | 2.7 | 2.4 | 3.1 | 2.5 | 1.8 | 2.1 | 1803 | 1771 | 1719 | 7 | 9 | 9 |

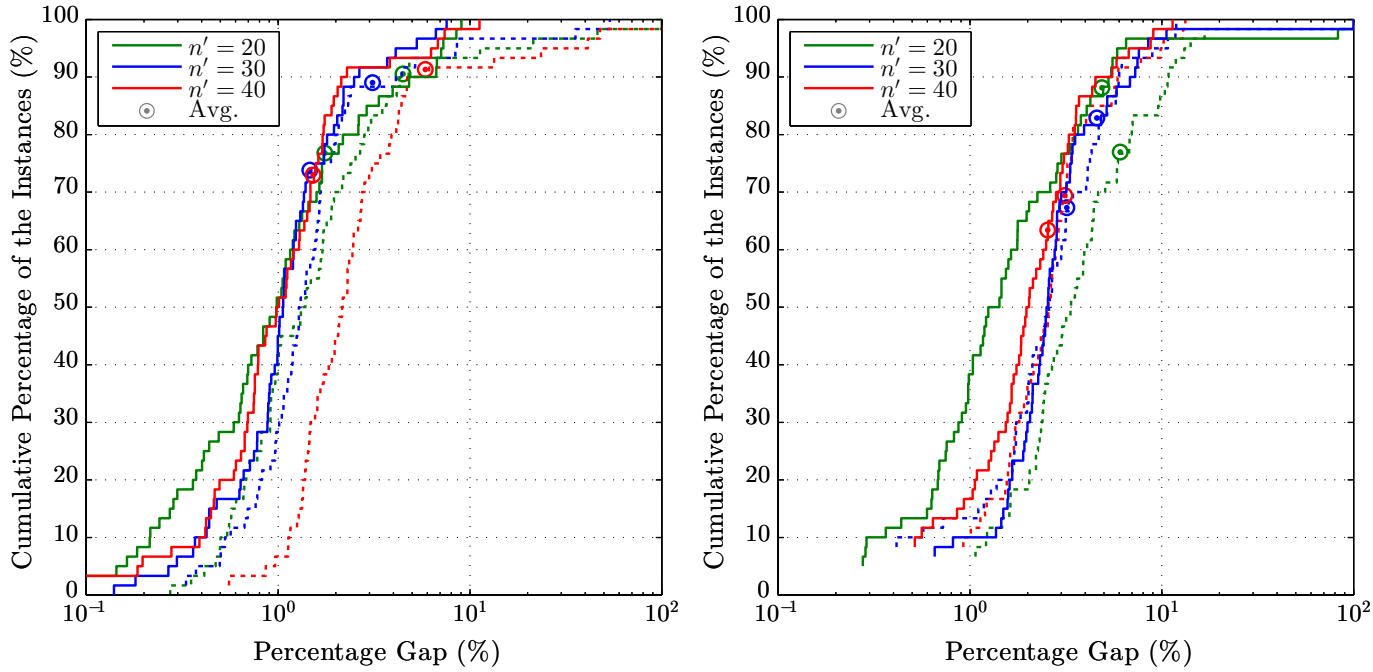
by our non-preemptive feasible solutions. For a given feasible solution, an upper bound on the optimality gap is calculated as $\frac{(\text{``OFV''}-\text{``Best Bound''})}{(\text{``Best Bound''})}$, where ``OFV'' is the objective function value of the feasible solution. We do not include detailed results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

The relatively higher gaps under "LB Quality" and "Feasible Sol'n" in Table 2 for TF = 0.4 stem from the small objective function values associated with loose due dates. Even small errors result in large percentage gaps in this case. Note that the objective function value of an instance with TF = 0.6, 0.8, and 1.0 is on average 7.5, 25.1, and 45.9 times larger, respectively, compared to that of an instance with TF = 0.4. A second contributing factor here is the growing size of (**TI**) with looser due dates. Frequently, even the LP relaxation is not solved within the allotted time for such instances, and this results in smaller "Best Bound" values in general. In other words, the actual performance for TF = 0.4 is probably better than what it appears to be.

The robustness of the quality of the feasible solutions obtained from our Benders-type approach is further illustrated in Figures 1a–1b. The empirical distributions of the optimality gaps of the feasible solutions associated with (**TR − A**) - **BDS** are plotted in these figures. The horizontal axes are in logarithmic scale to increase the readability of the graph. Note that the median percentage gap for each curve corresponds to the 50% mark on the vertical axis, and the average gaps are explicitly indicated. The curves are clustered and rise steeply. That is, the quality of the partitions retrieved from (**TR − A**) - **BDS** is not particularly sensitive to the increasing number of jobs $n'$ per machine.

The solution time performance of (**TR − A**) - **BDS** is overwhelmingly superior to that of (**TR − A**) - **CPX**. Based on the instances that are solved by both methods within the time limit, the ratio of the solution time of (**TR − A**) - **CPX** to that of (**TR − A**) - **BDS** is 46.7 on average. Out of a total of 720 instances, only 35 of them take slightly more time to solve for (**TR − A**) - **BDS** compared to (**TR − A**) - **CPX**. For both methods, instances with loose average due dates within a relatively wide range are more problematic. However, tightening the due dates does also hurt the performance of (**TR − A**) - **CPX** while it benefits that of (**TR − A**) - **BDS**. A detailed analysis of the empirical distributions of the solution times of (**TR − A**) - **BDS** and (**TR − A**) - **CPX** is available in Online Supplement J.1.

Recall that we call the SiPS/SiPSi libraries (Tanaka and Fujikuma, 2012, Tanaka et al., 2009) to solve $m$ single machine problems for each job partition present in the final solution pool of CPLEX and obtain feasible solutions for *Rm-TWT*




(a) $m$ = 2 (—) and 3 (– –)

(b) $m$ = 4 (—) and 5 (– –)

**Figure 1** The empirical distributions of the optimality gaps of the upper bounds by (**TR** − **A**) - **BDS** for *Rm-TWT*.

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

We conclude this section with a brief discussion on the time-indexed formulation (**TI**). The main purpose of solving (**TI**) in this paper is to incorporate the objective function value of the incumbent solution and the best lower bound available at termination into the `''Best Integer''` and `''Best Bound''` values, respectively, so that we quantify the gaps of our lower and upper bounds as accurately as possible. Otherwise, solving (**TI**) is not a scalable solution approach for *Rm-TWT* as we discussed previously in this section. In addition, the LP relaxation of (**TI**) does also suffer from the same scalability issue as a lower bounding method. We provide further specifics and settle this issue below.

On the one hand, the LP relaxations of time-indexed formulations are strong and provide very tight bounds. On the other hand, however, the size of a time-indexed formulation grows with the length of the planning horizon and is therefore pseudo-polynomial. From a computational perspective, the solution effort expended increases rapidly with longer processing times, and `CPLEX` cannot solve the LP relaxation of (**TI**) or find any feasible solution within the time limit of 1,800 seconds for the *Rm-TWT* (and *Rm-TWET*) instances with greater than 90 jobs. For the sake of completeness, we benchmarked the lower bound produced by (**TR** − **A**) - **BDS** against the optimal objective function value of the LP

relaxation of (**TI**). These two lower bounds do not dominate each other. There are instances in which the objective value of the LP relaxation of (**TI**) is larger than that of (**TR − A**) and vice versa. The best lower bound retrieved from (**TR − A**) - **BDS** at termination is on average 97.04% of the optimal objective value of the LP relaxation of (**TI**), computed over 360 *Rm-TWT* instances with $n \leq 90$. The corresponding figure for the *Rm-TWET* instances is 94.29%. Furthermore, recall that (**TR − A**) - **BDS** terminates with a 3% relative optimality gap. Therefore, it is fair to state that the lower bounds provided by (**TR − A**) - **BDS** and the LP relaxation of (**TI**) are of comparable quality. Ultimately, (**TR − A**) - **BDS** is the clear choice as a lower bounding technique given its superior computational time performance and the high quality of the non-preemptive schedules based on the solution of (**TR − A**) - **BDS**.

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

Initially, we also experimented with the identical parallel machine scheduling problems $Pm//\sum_j \pi_j T_j$ and $Pm//\sum_j \pi_j T_j + \epsilon_j E_j$. However, the symmetry inherent in these problems results in many similar cuts and causes (**TR − A**) - **BDS** to choke. One of the items in our future research agenda is exploring ways of enhancing our algorithm to be able to handle the identical parallel machine environment.

A further goal is to embed (**TR − A**) - **BDS** into an optimal algorithm for *Rm-TWT* and *Rm-TWET*. Note that the proposed preemptive relaxation can naturally handle branching decisions on the job to machine assignments.

**Acknowledgments.** We are grateful for the invaluable comments of the anonymous referees that helped us improve the paper.

**References**

Armentano, Vinícius A., Denise S. Yamashita. 2000. Tabu search for scheduling on identical parallel machines to minimize mean tardiness. *J Intell Manuf* **11** 453–460.

Azizoglu, Meral, Omer Kirca. 1998. Tardiness minimization on parallel machines. *Int J Prod Econ* **55** 163–168.

Azizoglu, Meral, Omer Kirca. 1999a. On the minimization of total weighted flow time with identical and uniform parallel machines. *Eur J Oper Res* **113** 91–100.

Azizoglu, Meral, Omer Kirca. 1999b. Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. *IIE Trans* **31** 153–159.

Baker, Kenneth R., Gary D. Scudder. 1990. Sequencing with earliness and tardiness penalties: A review. *Oper Res* **38** 22–36.

Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numer Math* **4** 238–252.

Biskup, Dirk, Jan Herrmann, Jatinder N.D. Gupta. 2008. Scheduling identical parallel machines to minimize total tardiness. *Int J Prod Econ* **115** 134–142.

Bülbül, Kerem, Philip Kaminsky, Candace Yano. 2007. Preemption in single machine earliness/tardiness scheduling. *J Sched* **10** 271–292.

Chen, Zhi-Long, Chung-Yee Lee. 2002. Parallel machine scheduling with a common due window. *Eur J Oper Res* **136** 512–527.

Chen, Zhi-Long, Warren B. Powell. 1999a. A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. *Eur J Oper Res* **116** 220–232.

Chen, Zhi-Long, Warren B. Powell. 1999b. Solving parallel machine scheduling problems by column generation. *INFORMS J Comput*

**11** 78–94.

Cheng, T.C.E., C.C.S. Sin. 1990. A state-of-the-art review of parallel-machine scheduling research. *Eur J Oper Res* **47** 271 – 292.

Detienne, Boris, Stéphane Dauzère-Pérès, Claude Yugma. 2011. Scheduling jobs on parallel machines to minimize a regular step total cost function. *J Sched* **14** 523–538.

Fischetti, Matteo, Domenico Salvagnin, Arrigo Zanette. 2010. A note on the selection of Benders' cuts. *Math Program* **124** 175–182.

Graham, R.L., E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. E.L. Johnson P.L. Hammer, B.H. Korte, eds., *Discrete Optimization II*, *Ann Discrete Math*, vol. 5. Elsevier, 287–326.

IBM ILOG CPLEX. 2011. IBM ILOG CPLEX Optimization Studio 12.4 Information Center. http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/index.jsp. Last viewed on 04/24/2013.

Jouglet, Antoine, David Savourey. 2011. Dominance rules for the parallel machine total weighted tardiness scheduling problem with release dates. *Comput Oper Res* **38** 1259–1266.

Kanet, John J., V. Sridharan. 2000. Scheduling with inserted idle time: Problem taxonomy and literature review. *Oper Res* **48** 99–110.

Kedad-Sidhoum, Safia, Yasmin Rios Solis, Francis Sourd. 2008. Lower bounds for the earliness–tardiness scheduling problem on parallel machines with distinct due dates. *Eur J Oper Res* **189** 1305–1316.

Koulamas, Christos. 1997. Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem. *Nav Res Log* **44** 109–125.

Lauff, Volker, Frank Werner. 2004. Scheduling with common due date, earliness and tardiness penalties for multimachine problems: A survey. *Math Comput Model* **40** 637–655.

Lenstra, J.K., A.H.G. Rinnooy Kan, P. Brucker. 1977. Complexity of machine scheduling problems. *Ann Discrete Math* **1** 343–362.

Liaw, Ching-Fang, Yang-Kuei Lin, Chun-Yuan Cheng, Mingchin Chen. 2003. Scheduling unrelated parallel machines to minimize total weighted tardiness. *Comput Oper Res* **30** 1777–1789.

Lin, Y.K., M.E. Pfund, J.W. Fowler. 2011. Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Comput Oper Res* **38** 901–916.

Luh, Peter B., Debra J. Hoitomt, Eric Max, Krishna R. Pattipati. 1990. Schedule generation and reconfiguration for parallel machines. *IEEE T Robotic Autom* **6** 687–696.

Magnanti, T. L., R. T. Wong. 1981. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Oper Res* **29** 464–484.

Mason, Scott J., Song Jin, Jagadish Jampani. 2009. A moving block heuristic to minimise earliness and tardiness costs on parallel machines. *Int J Prod Res* **47** 5377–5390.

M'Hallah, Rym, Talal Al-Khamis. 2012. Minimising total weighted earliness and tardiness on parallel machines using a hybrid heuristic. *Int J Prod Res* **50** 2639–2664.

Mönch, L. 2008. Heuristics to minimize total weighted tardiness of jobs on unrelated parallel machines. *2008 IEEE Inter Conf on Automation Science and Engineering*. IEEE, 572–577.

Pan, Yunpeng, Leyuan Shi. 2007. On the equivalence of the max-min transportation lower bound and the time-indexed lower bound for single-machine scheduling problems. *Math Program* **110** 543–559.

Pessoa, Artur, Eduardo Uchoa, Marcus Poggi Aragão, Rosiane Rodrigues. 2010. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Math Program Comput* **2** 259–290.

Pinedo, M. 2008. *Scheduling: Theory, Algorithms, and Systems*. 3rd ed. Springer.

Plateau, M.-C., Y. A. Rios-Solis. 2010. Optimal solutions for unrelated parallel machines scheduling problems using convex quadratic reformulations. *Eur J Oper Res* **201** 729–736.

Rios-Solis, Y. A., F. Sourd. 2008. Exponential neighborhood search for a parallel machine scheduling problem. *Comput Oper Res* **35** 1697–1712.

Rubin, Paul. 2011. Benders decomposition then and now. http://orinanobworld.blogspot.com/2011/10/benders-decomposition-then-and-now.html. Last viewed on 04/24/2013.

Şen, Halil, Kerem Bülbül. 2012. A simple, fast, and effective heuristic for the single-machine total weighted tardiness problem. Erik Demeulemeester, Willy Herroelen, eds., *Proceedings of the 13th Inter. Conf. on Project Management and Scheduling (PMS 2012)*. Leuven, Belgium, 282–286.

Sen, Tapan, Joanne M. Sulek, Parthasarati Dileepan. 2003. Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. *Int J Prod Econ* **83** 1 – 12.

Shim, Sang-Oh, Yeong-Dae Kim. 2007a. Minimizing total tardiness in an unrelated parallel-machine scheduling problem. *J Oper Res Soc* **58** 346–354.

Shim, Sang-Oh, Yeong-Dae Kim. 2007b. Scheduling on parallel identical machines to minimize total tardiness. *Eur J Oper Res* **177** 135–146.

Souayah, Nizar, Imed Kacem, Mohamed Haouari, Chengbin Chu. 2009. Scheduling on parallel identical machines to minimise the total weighted tardiness. *Inter J Adv Oper Manage* **1** 30–69.

Sourd, Francis, Safia Kedad-Sidhoum. 2003. The one-machine problem with earliness and tardiness penalties. *J Sched* **6** 533–549.

Tanaka, Shunji, Mituhiko Araki. 2008. A branch-and-bound algorithm with Lagrangian relaxation to minimize total tardiness on identical parallel machines. *Int J Prod Econ* **113** 446–458.

Tanaka, Shunji, Shuji Fujikuma. 2012. A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. *J Sched* **15** 347–361.

Tanaka, Shunji, Shuji Fujikuma, Mituhiko Araki. 2009. An exact algorithm for single-machine scheduling without machine idle time. *J Sched* **12** 575–593.

Üster, Halit, Homarjun Agrahari. 2011. A Benders decomposition approach for a distribution network design problem with consolidation and capacity considerations. *Oper Res Lett* **39** 138–143.

van den Akker, J. M., J. A. Hoogeveen, S. L. van de Velde. 1999. Parallel Machine Scheduling by Column Generation. *Oper Res* **47** 862–872.

Van Roy, Tony J. 1986. A cross decomposition algorithm for capacitated facility location. *Oper Res* **34** 145–163.

Wentges, Paul. 1996. Accelerating Benders' decomposition for the capacitated facility location problem. *Math Method Oper Res* **44** 267–290.

Yalaoui, Farouk, Chengbin Chu. 2002. Parallel machine scheduling to minimize total tardiness. *Int J Prod Econ* **76** 265–279.

Zhou, Hong, Zhengdao Li, Xuejing Wu. 2007. Scheduling Unrelated Parallel Machine to Minimize Total Weighted Tardiness Using Ant Colony Optimization. *2007 IEEE Inter Conf on Automation and Logistics*. IEEE, Jinan, 132–136.

## Appendix G. Online Supplement – Review of Related Literature

**G.1 Tardiness Heuristics for Identical Parallel Machines** Many of the heuristics developed for tardiness objectives on identical parallel machines apply list scheduling based on some priority index and sometimes enhance the initial schedule by local search. Yalaoui and Chu (2002) review several heuristics of this kind. An interesting deviation from the mainstream here is the decomposition heuristic by Koulamas (1997). The author heuristically extends the well-known decomposition principle valid for $1//\sum T_j$ to the problem $Pm//\sum T_j$ with very good results. At each iteration, the position of one job in the overall schedule is fixed, where the subproblems in the decomposition are solved by a fast and effective heuristic for $Pm//\sum T_j$ that observes the decomposition principle for the individual machine schedules. Furthermore, a hybrid simulated annealing heuristic is devised which is outperformed by the decomposition heuristic based on the solution quality and time trade-off. The results for 100-job instances indicate that the proposed heuristics are on average about 10-11% away from optimality with respect to a lower bound. A recent list scheduling heuristic by Biskup et al. (2008) for $Pm//\sum T_j$ yields somewhat better results than those of Koulamas for large instances with up to 5 machines and 200 jobs. An absolute assessment of the solution quality is not available due to the lack of a good lower bound or a scalable exact method. For the weighted version, i.e., the problem $Pm//\sum \pi_j T_j$, Armentano and Yamashita (2000) design a tabu search heuristic. For evaluation purposes, they benchmark their feasible solutions against the Lagrangian relaxation based lower bound by Luh et al. (1990). This lower bound is obtained by dualizing the machine capacity constraints in an integer programming formulation of the problem, similar to that by Tanaka and Araki (2008) discussed in the main text. In the original paper, Luh et al. include very limited computational experience, but the results of Armentano and Yamashita (2000) for instances with up to 10 machines and 150 jobs are promising. For instances with 100 jobs, the average optimality gap with respect to the Lagrangian lower bound of Luh et al. (1990) is 8.14% which drops to 5.80% for 150-job instances. On the flip side, Armentano and Yamashita report that computing the lower bound of Luh et al. takes about 3 hours for 100- and 150-job instances.

**G.2 Special Cases of *Rm-TWET*** Here we discuss studies that investigate special cases of the problem of scheduling a set of independent jobs on a bank of unrelated parallel machines with the objective of minimizing the total (weighted) earliness and tardiness. In Chen and Powell (1999a), a set partitioning model of the problem $Pm/d_j = d^l/\sum_j \epsilon_j E_j + \pi_j T_j$, where $d^l$ stands for an unrestrictively large common due date, is obtained through Dantzig-Wolfe reformulation. The linear programming (LP) relaxation of the set partitioning reformulation yields tight lower bounds, and instances with up to 60 jobs and 6 machines are solved to optimality. In a related study Chen and Lee (2002), the authors extend their approach by incorporating a common due date window and instances with up to 40 jobs and any number of machines are solved to optimality within reasonable times. Rios-Solis and Sourd (2008) consider the same problem as Chen and Powell (1999a), except that they allow for the common due date to be restrictively small. The main contribution of this work is a pseudo-polynomial time dynamic programming algorithm that can identify the best schedule in an exponential-size neighborhood of the current solution. Plateau and Rios-Solis (2010) is the only study available on common due date problems in the unrelated parallel machine environment that designs an optimal algorithm. The authors develop convex quadratic reformulations to solve both $Rm/d_j = d^l/\sum_j \epsilon_j E_j + \pi_j T_j$ and $Rm/d_j = d^r/\sum_j \epsilon_j E_j + \pi_j T_j$ exactly. For the first problem, the approach is successful. Instances with up to 4 machines and 50 jobs are solved optimally in at most one hour, and the bounds provided by the root relaxation are of high quality. However, the results for the latter restrictive case are not satisfactory. For further information and additional references on the common due date problems, the reader is referred to the survey paper by Lauff and Werner (2004) and the literature review in Rios-Solis and Sourd (2008).

**G.3 Overview of the Literature on *Rm-TWT* and *Rm-TWET***

**Table 3** Summary of the important points in Section 2.

| Paper | Problem | Method | Main Results. $[n, m]^{\dagger}$, Time/Gap$^{\dagger}$. |
|---|---|---|---|
| Liaw et al. (2003) | $Rm//\sum_j \pi_j T_j$ | Exact | B&B. First exact approach. [18, 4]. |
| Shim and Kim (2007a) | $Rm//\sum_j T_j$ | Exact | B&B. Lower bound of Liaw et al. (2003) and an alternate one. [20, 5], 60 min. |
| Zhou et al. (2007) | $Rm//\sum_j \pi_j T_j$ | Heuristic | Ant colony optimization. |
| Mönch (2008) | $Rm//\sum_j \pi_j T_j$ | Heuristic | Ant colony optimization. ATC dispatching, decomposition heuristic. |
| Lin et al. (2011) | $Rm//\sum_j \pi_j T_j$ | Heuristic | Genetic algorithm and two simple heuristics. [20, 4], 1.8%. |
| Plateau and Rios-Solis (2010) | $Rm/d_j = d/\sum_j \pi_j T_j + \epsilon_j E_j$ | Exact | Convex quadratic formulation. $d_j = d^r$ not satisfactory. [50, 4], 60 min. |
| Azizoglu and Kirca (1998) | $Pm//\sum_j T_j$ | Exact | B&B. Dominance rules. Lower bound based on minimal completion times. [15, 3]. |
| Yalaoui and Chu (2002) | $Pm//\sum_j T_j$ | Exact | B&B. [20, 2], 30 min. |
| Shim and Kim (2007b) | $Pm//\sum_j T_j$ | Exact | B&B. Dominance rules. [30, 5], 60 min. |
| Tanaka and Araki (2008) | $Pm//\sum_j T_j$ | Exact | Lagrangian relaxation to time-indexed formulation. [25, 10]. |
| Souayah et al. (2009) | $Pm//\sum_j \pi_j T_j$ | Exact | Mix of lower bounds. [35, 2], 20 min. |
| Pessoa et al. (2010) | $Pm//\sum_j \pi_j T_j$ | Exact | Branch-cut-and-price. Arc-time-indexed formulation. Best to date. [100, 4]. |
| Jouglet and Savourey (2011) | $Pm/r_j/\sum_j \pi_j T_j$ | Exact | B&B. Dominance rules. [20, 3]. |
| Koulamas (1997) | $Pm//\sum_j T_j$ | Heuristic | Decomposition heuristic. [100, 8], ~ 10%. |
| Armentano and Yamashita (2000) | $Pm//\sum_j \pi_j T_j$ | Heuristic | Tabu search. Lagrangian relaxation of Luh et al. (1990). [150, 10], 5%-10%. |
| Biskup et al. (2008) | $Pm//\sum_j T_j$ | Heuristic | List scheduling. [200, 5]. |
| Chen and Powell (1999a) | $Pm/d_j = d^l/\sum_j \pi_j T_j + \epsilon_j E_j$ | Exact | Set partitioning formulation. Dantzig-Wolfe decomposition. [60, 6]. |
| Chen and Lee (2002) | $Pm/[d_1, d_2]/\sum_j \pi_j T_j + \epsilon_j E_j$ | Exact | Extends Chen and Powell (1999a). Common due window. Column generation. [40, $m$]. |
| Kedad-Sidhoum et al. (2008) | $Pm/r_j/\sum_j \pi_j T_j + \epsilon_j E_j$ | Heuristic | Lagrangian relaxation to time-indexed formulation. [90, 6], 1.5%. |
| Rios-Solis and Sourd (2008) | $Pm/d_j = d^r/\sum_j \pi_j T_j + \epsilon_j E_j$ | Heuristic | DP algorithm to explore exponential-size neighborhood. |
| Mason et al. (2009) | $Pm//\sum_j T_j + E_j$ | Heuristic | Moving-block heuristic. [40, 4]. |
| M'Hallah and Al-Khamis (2012) | $Pm//\sum_j \pi_j T_j + \epsilon_j E_j$ | Heuristic | Two constructive and three meta-heuristics. [90, 6], 1.4%-6.4%. |
| Cheng and Sin (1990) | "A State-of-the-Art Review of Parallel-Machine Scheduling" | | |
| Baker and Scudder (1990) | "Sequencing with Earliness and Tardiness Penalties: A Review" | | |
| Kanet and Sridharan (2000) | "Scheduling with Inserted Idle Time: Problem Taxonomy and Literature Review" | | |
| Sen et al. (2003) | "Static Scheduling Research to Minimize Weighted and Unweighted Tardiness: A State-of-the-Art Survey" | | |
| Lauff and Werner (2004) | "Scheduling with Common Due Date, Earliness and Tardiness Penalties for Multimachine Problems: A Survey" | | |

$^{\dagger}$: Largest instance size tackled successfully and the associated time / optimality gap information if available.

## Appendix H. Online Supplement – Proofs

**H.1 Proof of Proposition 3.1** Proof. Let $S_{\mathbf{P}}$ represent a feasible schedule for problem (**P**) with a total cost of $TC\left(S_{\mathbf{P}}\right)$. The notation (**P**($\overline{\boldsymbol{y}}$)) stands for problem (**P**) in which the jobs are assigned to the machines a priori, but the individual machine schedules for this job partition $\overline{\boldsymbol{y}}$ are to be optimized. An optimal schedule is denoted by an asterisk in the superscript.

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

$$TC\left(S^*_{\mathbf{TR-A}(\overline{\boldsymbol{y}})}\right) = \sum_{k=1}^{m} TC\left(S^*_{\mathbf{TR-A}(\overline{\boldsymbol{y}}_k)}\right) \leq \sum_{k=1}^{m} TC\left(S^*_{\mathbf{NP}(\overline{\boldsymbol{y}}_k)}\right) = TC\left(S^*_{\mathbf{NP}(\overline{\boldsymbol{y}})}\right).$$

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

**H.2 Proof of Lemma 4.1** Proof. Assume that an optimal solution $(\overline{\boldsymbol{u}}_k, \overline{\boldsymbol{v}}_k)$ to $(\mathbf{DS_k} - \mathbf{R})$ is available. The claim holds trivially if there are idle periods in the schedule – which would typically be true for an instance of *Rm-TWET* – because for any idle period $t$ we have $\overline{v}_{kt} = 0$ due to complementary slackness. We set $(\overline{\boldsymbol{u}}'_k, \overline{\boldsymbol{v}}'_k) = (\overline{\boldsymbol{u}}_k, \overline{\boldsymbol{v}}_k)$.

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

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

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

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

Therefore, $(\overline{\boldsymbol{u}}'_k, \overline{\boldsymbol{v}}'_k)$ is an alternate optimal solution, and $\max_{t=1,\ldots,H_k} \overline{v}'_{kt} = \max_{t=1,\ldots,H_k} \left\{ \overline{v}_{kt} + \mid \overline{v}^{\max}_k \mid \right\} = 0$ by the definition of $\overline{v}^{\max}_k$. □

## Appendix I. Online Supplement – Benders Decomposition Algorithm with Cut Strengthening for (TR − A)

**Algorithm 1:** Solving (**TR** − **A**) by Benders decomposition and lazy constraint generation.

`// Initialization`

Create (**RMP**) with (19), (20), (22). Add the load balancing constraints (14) for *Rm-TWT*;

**repeat** `// To improve the initial objective value of (RMP).`

  Construct a feasible assignment $\overline{y}$ of jobs to $m$ machines by some heuristic.

  $[cuts, z_1(\overline{y}), \ldots, z_m(\overline{y})] = generate\_cuts(\overline{y})$ ; `// cuts is a collection of m cuts.`

  Add *cuts* to (**RMP**) as lazy constraints;

**until** *some termination condition is satisfied*; `// We run a simple dispatch rule once.`

`// Main loop`

Invoke CPLEX on (**RMP**);

**repeat**

  Identify a new candidate incumbent solution $\overline{y}$ with an objective value of $\sum_{k=1}^{m} \overline{\eta}_k$;

  *accept_candidate* = true;

  $[cuts, z_1(\overline{y}), \ldots, z_m(\overline{y})] = generate\_cuts(\overline{y})$ ; `// cuts is a collection of m cuts.`

  **for** $k = 1$ *to* $m$ **do**

    **if** $\overline{\eta}_k < z_k(\overline{y})$ **then** `// y violates some of the missing Benders cuts.`

      Add $cuts_k$ to (**RMP**) as a lazy constraint, *accept_candidate* = false;

  **end**

**until** *CPLEX determines that the relative optimality gap of the current incumbent is less than some threshold*;

The best available job partition $\overline{y}^*$ for (**TR** − **A**) is retrieved from CPLEX. If desired, the associated preemptive machine schedules are obtained by solving ($\mathbf{TR}_k$ − **R**) with $\overline{y}^*$ for $k = 1, \ldots, m$.

**Algorithm 2:** Procedure *generate_cuts*.

**input** : A feasible partition $\overline{y}$ of jobs to machines.

**output**: Returns $z_k(\overline{y})$ and the strengthened cuts of the form (24) for $k = 1, \ldots, m$.

**for** $k = 1$ *to* $m$ **do**

  Solve ($\mathbf{TR}_k$ − **R**), retrieve $z_k(\overline{y})$ and the optimal solution $(\overline{u}_k, \overline{v}_k)$ for the dual slave ($\mathbf{DS}_k$ − **R**);

  `/* Calculate an alternate optimal solution` $(\overline{u}'_k, \overline{v}'_k)$ `for` ($\mathbf{DS}_k$ − **R**) `that satisfies Lemma 4.1 by following the construction in the proof. */`

  $\overline{v}_k^{\max} = \max_{t=1,\ldots,H_k} \overline{v}_{kt}$;

  **if** $\overline{v}_k^{\max} < 0$ **then** $\overline{u}'_{jk} = \overline{u}_{jk} - |\overline{v}_k^{\max}|$, $j \in J_k$, $\overline{v}'_{kt} = \overline{v}_{kt} + |\overline{v}_k^{\max}|$, $t = 1, \ldots, H_k$ **else** $(\overline{u}'_k, \overline{v}'_k) = (\overline{u}_k, \overline{v}_k)$;

  `// Construct an optimal solution` $(\overline{u}''_k, \overline{v}''_k)$ `for` ($\mathbf{DS}_k$).

  $\overline{v}''_{kt} = \overline{v}'_{kt}$, $t = 1, \ldots, H_k$, and $\overline{v}''_{kt} = 0$, $t = H_k + 1, \ldots, H$;

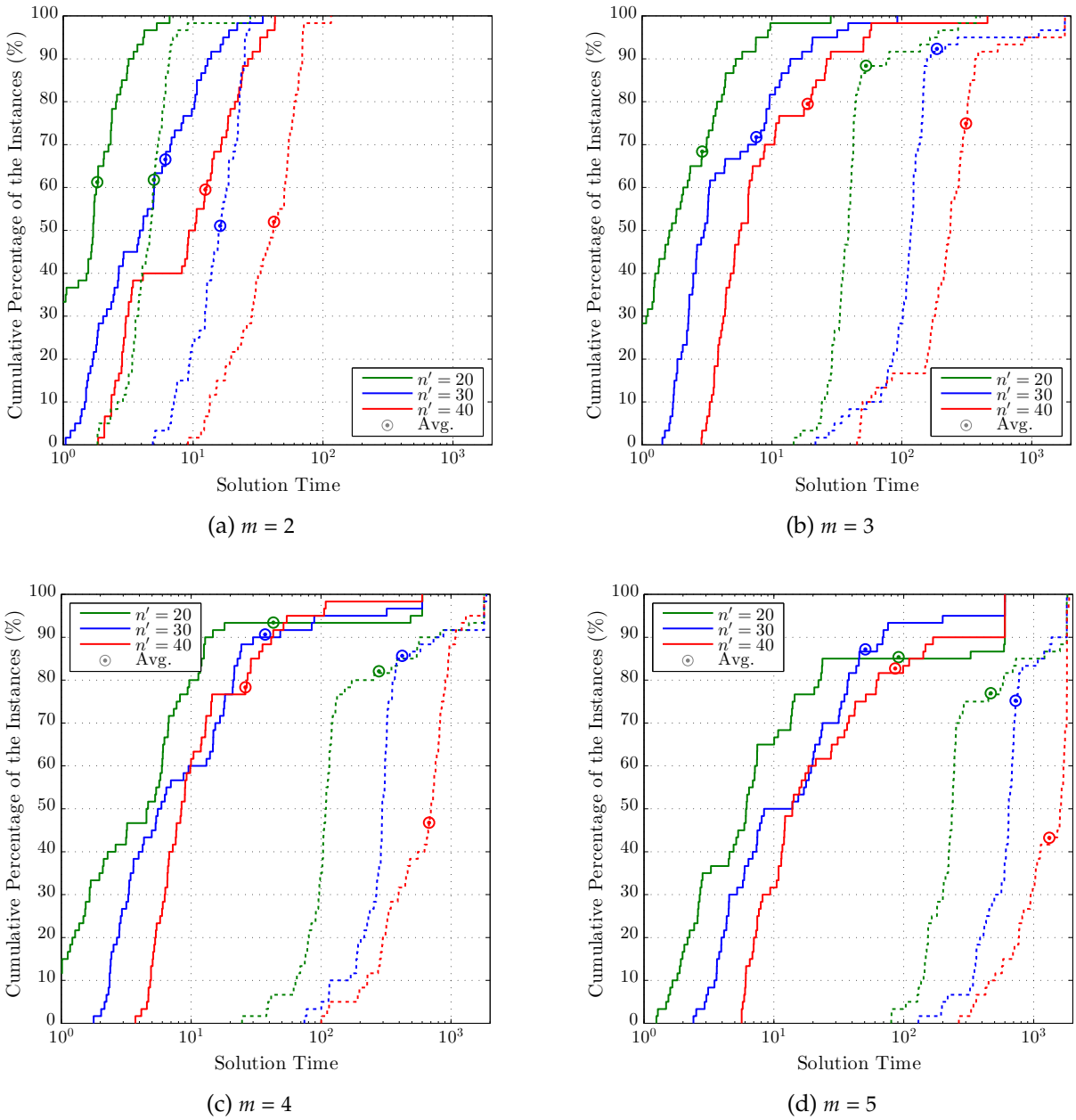
  $\overline{u}''_{jk} = \overline{u}'_{jk}$, $j \in J_k$, and $\overline{u}''_{jk}$, $j \notin J_k$, is calculated based on either (29) or (30), respectively, depending on whether we solve an instance of *Rm-TWT* or *Rm-TWET*;

  Generate and add (24) to *cuts*;

**end**

## Appendix J. Online Supplement – Computational Results

**J.1 Analysis of the Solution Times of (TR − A) - BDS and (TR − A) - CPX for *Rm-TWT*** The empirical distributions of the solution times of (**TR − A**) - **BDS** and (**TR − A**) - **CPX** are plotted with solid and dashed lines in Figure 2, respectively. Similar to those in Figure 1, the horizontal axes are in logarithmic scale. The performance of (**TR − A**) - **CPX** is adversely affected by both an increasing number of machines $m$ and an increasing number of jobs per machine $n'$ in an instance. To make the former observation concrete, note that the percentage of the instances with $n' = 20$ solved to optimality by (**TR − A**) - **CPX** within 60 seconds is 100%, 88.3%, 6.7%, and 0% for $m = 2, 3, 4, 5$, respectively. In comparison, (**TR − A**) - **BDS** obtains the optimal solution for 100%, 98.9%, 93.3%, and 82.8% of the instances with $m = 2, 3, 4, 5$, respectively, in less than 60 seconds. Note that these latter numbers are aggregated over $n'$, including larger instances with $n' = 30, 40$ as well. Clearly, (**TR − A**) - **BDS** displays a significantly more stable performance. Finally, we note that the solution times of (**TR − A**) - **BDS** are strongly correlated with the number of Benders cuts generated, as expected. The median percentage of the active Benders cuts for the final node problem in the search tree is 86.4% with a corresponding average of 81.3%.




(a) $m = 2$ (b) $m = 3$

(c) $m = 4$ (d) $m = 5$

**Figure 2** The empirical distributions of the solution times of (**TR − A**) - **BDS** and (**TR − A**) - **CPX** for *Rm-TWT*.

**J.2 Results for *Rm-TWET*** Table 4 is structured identically to Table 2 and depicts the percentage gap and solution time results for *Rm-TWET*. Unsurprisingly, both solving the preemptive relaxation and obtaining high-quality non-preemptive solutions pose a more difficult challenge for *Rm-TWET* than for *Rm-TWT*. In general, the gaps are larger and the solution times are longer than those in Section 5.1. However, in the grand scheme of things – also factoring in the lack of scalable alternate algorithms for this problem in the literature – we attain pretty promising results for *Rm-TWET* as well.

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

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

The results on the optimality gaps of the non-preemptive solutions included under "Feasible Sol'n" in Table 4 certify (**TR** − **A**) - **BDS** as a viable and scalable algorithm for solving large instances of *Rm-TWET*. As is the case with *Rm-TWT*, even the LP relaxation of (**TI**) is not solved within half an hour for instances with 100 or more jobs. Among the smaller 360 instances, (**TI**) beats (**TR** − **A**) - **BDS** in 125 cases with an average improvement of 0.84%. For the other 235 instances, (**TR** − **A**) - **BDS** outperforms (**TI**) by 40.17% on average. The optimality gap of the incumbent from (**TI**) is over 15% in 39% of these instances (142 instances) while (**TR** − **A**) - **BDS** does always keep the gap below the same threshold with the exception of 5 instances. Even for the 360 larger instances with 100 or more jobs, the proposed Benders-type method finds a feasible solution for the original problem with an optimality gap less than 15% in 86% of the cases (310 instances). The behavior of (**TR** − **A**) - **BDS** with respect to the varying TF and RDD levels in Table 4 is consistent with our observations for Table 2. The adverse effect of low TF and high RDD values on both the lower and upper bound quality persists with the same underlying reasons explained in Section 5.1.

**Table 4** Results for *Rm-TWET*.

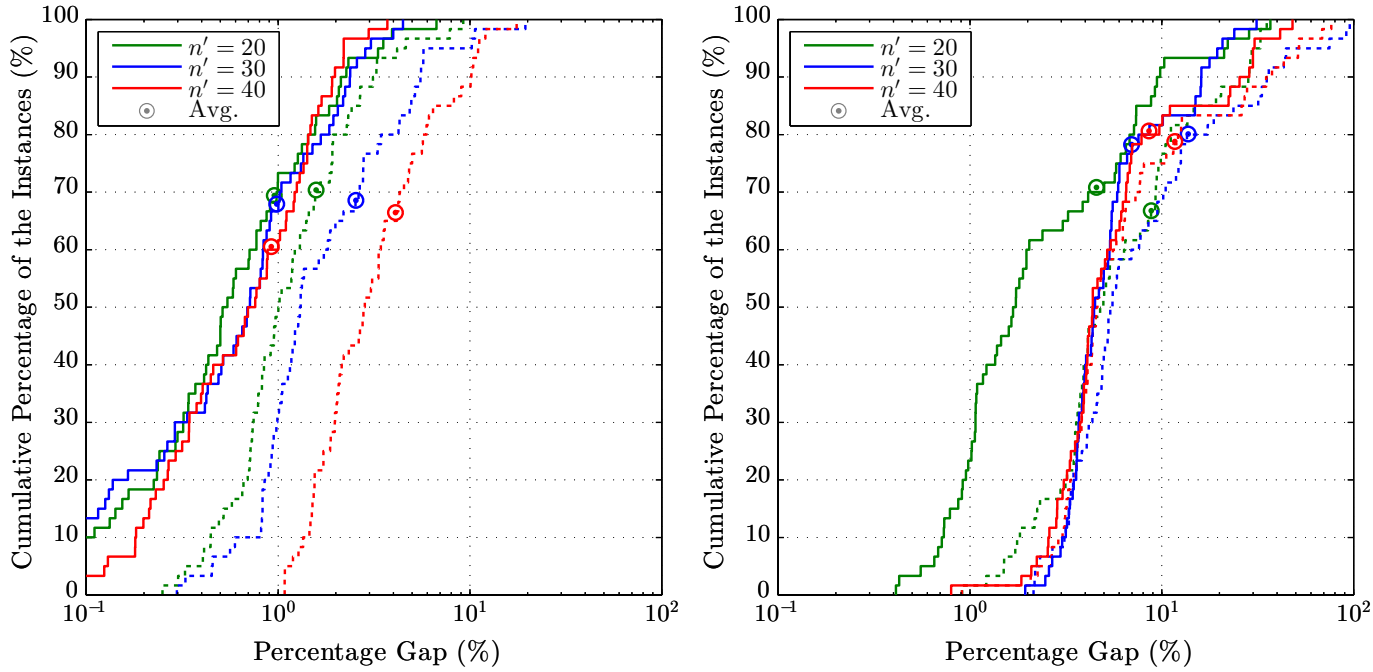
| | | RDD | Percentage Gaps | | | | | | | | | (**TR** − **A**) - Solution Times | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | (**TR** − **A**) - **BDS** | | | LB Quality | | | Feasible Sol'n | | | **CPX** | | | **BDS** | | |
| $n$ | $m$ | TF | 0.2 | 0.4 | 0.6 | 0.2 | 0.4 | 0.6 | 0.2 | 0.4 | 0.6 | 0.2 | 0.4 | 0.6 | 0.2 | 0.4 | 0.6 |
| 40 | 2 | 0.4 | 2.7 | 2.9 | 4.4 | 4.7 | 6.9 | 15.1 | 0.8 | 0.9 | 3.2 | 15 | 13 | 21 | 16 | 22 | 454 |
| | | 0.6 | 2.6 | 2.8 | 2.9 | 4.7 | 7.4 | 10.3 | 0.8 | 2.0 | 0.7 | 16 | 12 | 14 | 14 | 21 | 48 |
| | | 0.8 | 2.0 | 2.3 | 2.8 | 3.1 | 4.6 | 5.7 | 1.0 | 0.8 | 0.6 | 17 | 15 | 14 | 10 | 10 | 17 |
| | | 1.0 | 1.9 | 1.4 | 1.5 | 2.2 | 2.2 | 2.7 | 0.3 | 0.2 | 0.2 | 16 | 16 | 15 | 5 | 6 | 8 |
| 60 | 2 | 0.4 | 2.1 | 3.0 | 5.4 | 3.8 | 6.6 | 13.7 | 0.7 | 1.1 | 2.7 | 59 | 43 | 47 | 47 | 146 | 550 |
| | | 0.6 | 2.2 | 2.7 | 3.0 | 3.9 | 6.8 | 10.0 | 0.7 | 1.7 | 2.6 | 62 | 48 | 45 | 41 | 52 | 249 |
| | | 0.8 | 1.8 | 1.8 | 2.6 | 2.9 | 3.8 | 5.7 | 0.4 | 0.6 | 0.8 | 65 | 56 | 53 | 24 | 35 | 72 |
| | | 1.0 | 1.6 | 0.6 | 1.7 | 1.8 | 1.1 | 2.8 | 0.2 | 0.1 | 0.3 | 69 | 71 | 70 | 14 | 23 | 26 |

Continued on next page...

Table 4 continued...

| $n$ | $m$ | RDD \ TF | Percentage Gaps: (**TR – A**) - **BDS** 0.2 | 0.4 | 0.6 | LB Quality 0.2 | 0.4 | 0.6 | Feasible Sol'n 0.2 | 0.4 | 0.6 | (**TR – A**) - Solution Times: **CPX** 0.2 | 0.4 | 0.6 | **BDS** 0.2 | 0.4 | 0.6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 80 | 3 | 0.4 | 2.6 | 3.0 | 10.1 | 5.3 | 8.1 | 17.4 | 1.5 | 1.3 | 2.3 | 121 | 219 | 384 | 17 | 207 | 601 |
| | | 0.6 | 2.9 | 2.9 | 6.0 | 5.0 | 7.9 | 17.9 | 1.4 | 2.3 | 5.3 | 199 | 183 | 382 | 6 | 20 | 541 |
| | | 0.8 | 2.0 | 2.6 | 2.9 | 3.4 | 5.7 | 7.0 | 0.9 | 0.8 | 1.1 | 256 | 216 | 211 | 3 | 9 | 18 |
| | | 1.0 | 2.4 | 2.4 | 2.6 | 2.4 | 2.7 | 4.3 | 0.7 | 0.7 | 0.8 | 231 | 237 | 232 | 2 | 2 | 5 |
| | 2 | 0.4 | 2.1 | 2.8 | 6.6 | 3.5 | 5.6 | 13.3 | 0.9 | 1.1 | 2.6 | 198 | 156 | 118 | 68 | 142 | 588 |
| | | 0.6 | 2.0 | 2.7 | 3.1 | 3.1 | 5.4 | 7.9 | 0.5 | 1.5 | 1.7 | 194 | 140 | 143 | 47 | 74 | 328 |
| | | 0.8 | 1.9 | 1.7 | 2.6 | 2.6 | 3.1 | 5.0 | 0.4 | 0.5 | 1.0 | 210 | 198 | 155 | 35 | 51 | 70 |
| | | 1.0 | 1.9 | 1.0 | 1.0 | 2.0 | 1.5 | 1.9 | 0.3 | 0.2 | 0.3 | 227 | 224 | 204 | 19 | 33 | 39 |
| | 4 | 0.4 | 2.9 | 5.9 | 31.2 | 5.8 | 12.1 | 50.7 | 2.8 | 4.9 | 20.6 | 433 | 647 | 1708 | 48 | 596 | 609 |
| | | 0.6 | 2.7 | 3.6 | 20.3 | 6.4 | 9.7 | 35.6 | 2.8 | 4.6 | 11.5 | 604 | 611 | 1947 | 23 | 324 | 605 |
| | | 0.8 | 2.4 | 2.8 | 3.0 | 4.2 | 6.2 | 8.2 | 1.1 | 1.9 | 1.5 | 628 | 526 | 644 | 17 | 31 | 426 |
| | | 1.0 | 2.8 | 2.1 | 2.7 | 3.1 | 3.1 | 4.6 | 1.0 | 0.9 | 1.4 | 773 | 685 | 680 | 10 | 8 | 18 |
| 90 | 3 | 0.4 | 2.7 | 3.7 | 15.2 | 4.6 | 8.3 | 27.3 | 1.9 | 3.3 | 8.6 | 439 | 495 | 920 | 16 | 335 | 606 |
| | | 0.6 | 2.1 | 2.8 | 6.6 | 4.0 | 6.8 | 15.7 | 1.2 | 3.0 | 5.3 | 721 | 595 | 757 | 13 | 120 | 605 |
| | | 0.8 | 2.8 | 2.7 | 2.6 | 3.9 | 5.0 | 5.9 | 0.8 | 1.6 | 1.9 | 800 | 716 | 578 | 10 | 11 | 43 |
| | | 1.0 | 2.7 | 2.4 | 2.2 | 2.7 | 3.4 | 3.4 | 1.0 | 0.9 | 1.0 | 739 | 691 | 669 | 8 | 10 | 11 |
| 100 | 5 | 0.4 | 3.1 | 9.6 | 34.6 | 6.1 | 16.9 | 57.9 | 4.2 | 10.5 | 30.9 | 1555 | 1805 | 1849 | 394 | 611 | 606 |
| | | 0.6 | 2.9 | 6.5 | 22.2 | 6.1 | 14.1 | 41.5 | 4.5 | 10.0 | 22.9 | 1498 | 1670 | 1804 | 99 | 601 | 600 |
| | | 0.8 | 2.6 | 3.0 | 6.2 | 4.7 | 6.5 | 12.0 | 3.4 | 5.1 | 7.4 | 1363 | 1111 | 1399 | 8 | 109 | 490 |
| | | 1.0 | 2.5 | 1.9 | 2.8 | 3.0 | 3.1 | 5.1 | 1.4 | 2.1 | 3.5 | 1614 | 1480 | 1382 | 3 | 4 | 16 |
| 120 | 3 | 0.4 | 2.4 | 3.0 | 11.5 | 3.8 | 6.6 | 20.9 | 2.5 | 5.0 | 12.0 | 1331 | 1053 | 978 | 33 | 376 | 601 |
| | | 0.6 | 1.9 | 2.9 | 5.0 | 3.2 | 6.5 | 12.8 | 2.0 | 4.9 | 9.2 | 1448 | 1098 | 1037 | 15 | 136 | 601 |
| | | 0.8 | 2.7 | 2.2 | 2.8 | 3.6 | 4.1 | 5.5 | 1.5 | 2.9 | 3.9 | 1679 | 1416 | 1103 | 9 | 13 | 55 |
| | | 1.0 | 2.7 | 2.4 | 2.2 | 2.9 | 3.0 | 3.2 | 1.3 | 2.0 | 1.8 | 1699 | 1726 | 1481 | 7 | 7 | 11 |
| | 4 | 0.4 | 2.9 | 4.3 | 24.4 | 5.1 | 8.6 | 41.4 | 3.8 | 5.9 | 22.6 | 1710 | 1568 | 1819 | 64 | 601 | 601 |
| | | 0.6 | 2.6 | 3.5 | 12.2 | 4.6 | 8.6 | 24.8 | 4.3 | 6.5 | 15.6 | 1805 | 1691 | 1789 | 19 | 311 | 601 |
| | | 0.8 | 2.4 | 2.8 | 3.2 | 3.8 | 5.4 | 7.4 | 3.8 | 4.8 | 6.3 | 1812 | 1802 | 1692 | 9 | 56 | 244 |
| | | 1.0 | 2.7 | 2.5 | 2.3 | 3.2 | 3.4 | 3.9 | 3.0 | 3.4 | 3.6 | 1812 | 1806 | 1803 | 5 | 7 | 10 |
| 150 | 5 | 0.4 | 2.9 | 8.3 | 30.4 | 4.9 | 14.8 | 66.9 | 4.9 | 14.8 | 66.0 | 1895 | 1865 | 2093 | 154 | 601 | 602 |
| | | 0.6 | 2.9 | 4.8 | 17.3 | 5.2 | 10.8 | 36.0 | 5.2 | 10.8 | 36.0 | 1843 | 1986 | 1859 | 46 | 601 | 601 |
| | | 0.8 | 2.4 | 2.9 | 3.8 | 4.0 | 5.7 | 8.3 | 4.0 | 5.7 | 8.3 | 1858 | 1858 | 1850 | 13 | 83 | 507 |
| | | 1.0 | 2.5 | 1.9 | 2.1 | 3.0 | 2.8 | 3.6 | 3.0 | 2.8 | 3.6 | 1900 | 1822 | 1812 | 7 | 10 | 12 |
| 160 | 4 | 0.4 | 2.6 | 4.3 | 18.1 | 3.9 | 8.5 | 35.1 | 3.9 | 8.5 | 35.1 | 1880 | 1850 | 1819 | 78 | 489 | 601 |
| | | 0.6 | 2.1 | 3.0 | 10.6 | 3.6 | 6.6 | 22.7 | 3.6 | 6.6 | 22.7 | 1876 | 1836 | 1880 | 23 | 200 | 601 |
| | | 0.8 | 2.6 | 2.6 | 2.8 | 3.7 | 4.6 | 5.8 | 3.7 | 4.6 | 5.8 | 1907 | 1841 | 1843 | 15 | 41 | 138 |
| | | 1.0 | 2.1 | 2.6 | 1.8 | 2.5 | 3.3 | 3.0 | 2.3 | 3.3 | 3.0 | 1886 | 1844 | 1836 | 10 | 11 | 18 |
| 200 | 5 | 0.4 | 2.9 | 6.9 | 28.3 | 4.4 | 11.9 | 58.4 | 4.4 | 11.9 | 58.4 | | | | 127 | 601 | 601 |
| | | 0.6 | 2.1 | 2.9 | 14.9 | 3.5 | 7.2 | 30.7 | 3.5 | 7.2 | 30.7 | | | | 28 | 325 | 601 |
| | | 0.8 | 2.0 | 2.8 | 3.0 | 3.0 | 4.7 | 6.4 | 3.0 | 4.7 | 6.4 | | | | 22 | 81 | 378 |
| | | 1.0 | 2.3 | 2.8 | 2.7 | 2.7 | 3.6 | 3.9 | 2.7 | 3.6 | 3.9 | | | | 11 | 13 | 19 |

Figure 3 is the counterpart of Figure 1, where the empirical distributions of the optimality gaps of the feasible solutions associated with (**TR – A**) - **BDS** are plotted. As previously, (**TR – A**) - **BDS** generally exhibits a robust behavior with respect to varying values of $n'$ for a fixed $m$. The problem gets more challenging with an increasing number of machines,




(a) $m = 2$ (—) and 3 (– –)

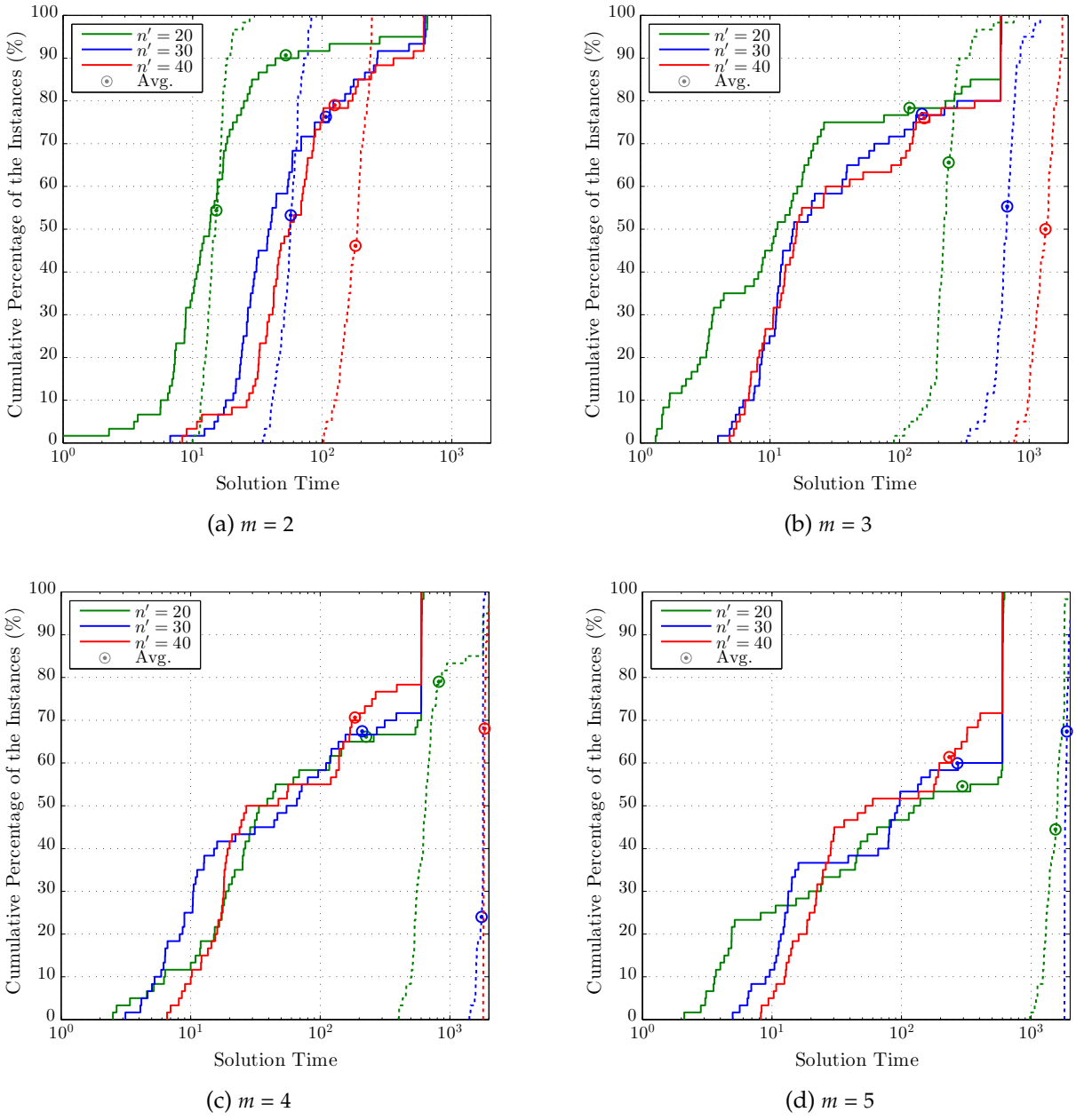
(b) $m = 4$ (—) and 5 (– –)

**Figure 3** The empirical distributions of the optimality gaps of the upper bounds by (**TR** − **A**) - **BDS** for *Rm-TWET*.

and the percentage gaps associated with (**TR** − **A**) - **BDS** demonstrate a modest increase with increasing $m$. For instance, for 70% of the instances with 2, 3, 4, and 5 machines, the gaps are less than 2%, 5%, 7%, and 11%, respectively.

The performance patterns observed for *Rm-TWT* pretty much carry over to *Rm-TWET* as well. The solution times of (**TR** − **A**) - **BDS** are in general better than those of (**TR** − **A**) - **CPX** by a large margin. Based on the 399 instances that are solved by both methods within their respective time limits, the ratio of the solution time of (**TR** − **A**) - **CPX** to that of (**TR** − **A**) - **BDS** is 48.0 on average. Among these instances, only 52 of the relatively smaller instances with less than 90 jobs and generally large RDD values take longer for (**TR** − **A**) - **BDS**. As in Table 2, low TF and high RDD values result in tough instances to handle for (**TR** − **A**) - **BDS** while instances with tight due dates are solved extremely well. The empirical distributions of the solution times of (**TR** − **A**) - **BDS** and (**TR** − **A**) - **CPX**, plotted with solid and dashed lines in Figure 4, respectively, reveal that the median solution times of (**TR** − **A**) - **BDS** are in the range from 11 to 125 seconds for all $m, n'$ combinations. (**TR** − **A**) - **CPX** features a much less robust behavior with a median solution time of 15 seconds for $n' = 20$ and $m = 2$ that quickly increases to 220, 649, and 1589 seconds for $n' = 20$ and $m = 3, 4, 5$, respectively. Compared to those in Table 2, the computational effort expended is significantly more. To be specific, the median solution times of (**TR** − **A**) - **CPX** for the *Rm-TWET* instances with 2, 3, 4, and 5 machines are 5, 7, 7, and 4 times of those for the corresponding *Rm-TWT* instances, respectively. The respective ratios for (**TR** − **A**) - **BDS** are 11, 4, 6, and 8. The greater planning horizons in the formulations are a primary factor here in addition to the inherent difficulty of *Rm-TWET* over *Rm-TWT*. This difficulty is also reflected in the number of Benders cuts generated. (**TR** − **A**) - **BDS** needs to create 5.7 times more cuts for *Rm-TWET* compared to *Rm-TWT*, and the great majority of these cuts is not redundant. The median percentage of the active Benders cuts for the final node problem in the search tree is 95.5% with a corresponding average of 92.1%. Note that these numbers are higher than their counterparts for *Rm-TWT*.

The times expended to solve the single machine problems for a given job partition – omitted from Table 4 for the sake of brevity – are more than satisfactory. The `SiPS/SiPSi` solver returns the optimal solution for a single machine TWET problem in about 27, 110, and 305 milliseconds for instances with $n' = 20, 30, 40$, respectively. These numbers translate into 23 seconds on average to solve all single machine problems to optimality for a five machine and 200 job instance




(a) $m = 2$ (b) $m = 3$

(c) $m = 4$ (d) $m = 5$

**Figure 4** The empirical distributions of the solution times of (**TR − A**) - **BDS** and (**TR − A**) - **CPX** for *Rm-TWET*.

with a maximum of 56 seconds. While these figures are greater than their counterparts for *Rm-TWT*, they still make up for a small part of the total solution time. The time spent for calculating the non-preemptive solutions accounts for only 12.5% of the total solution time on average. Finally, we note that (**TR − A**) - **BDS** identifies on average 5.4 times more job partitions per instance compared to (**TR − A**) - **CPX**, where the average number of partitions retrieved from the search tree of (**TR − A**) - **BDS** is 14.4. As we discussed in Section 5.1, this is a critical advantage that improves the quality of the best non-preemptive solution.