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
May 12, 2013

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 fast and effective mathematical programming based heuristics 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 ϵ_j and π_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. 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 (1999b), and Azizoglu and Kirca (1999a). 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. We will delve into this discussion further when we introduce our preemptive relaxation in Section 3.

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. For the same problem, 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. Tanaka and Araki (2008) take a different path and apply Lagrangian relaxation to the time-indexed formulation of the problem $Pm//\sum T_j$ in an effort to develop tighter 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 $Pm//\sum_j w_j T_j$. 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. The study by Pessoa et al. (2010) is by far the most successful exact algorithm to date. A branch-cut-and-price algorithm applied to an arc-time-indexed formulation delivers optimal solutions to instances of $Pm//\sum w_j T_j$ with up to 100 jobs and 4 machines. In another recent study, Detienne et al. (2011) investigate the problem $Rm/r_j/\sum_j \overline{f}_j(C_j)$, where $\overline{f}_j(C_j)$ represents a stepwise cost function associated with job j. This problem subsumes *Rm-TWT*; however, the focus of

the authors is on job cost functions with a few steps. For the tardiness objective, the first jump occurs at the due date with subsequent jumps at every time point. 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 beyond reach for the existing exact methods. Several authors (see Section 3) attribute this fact 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 parallel machines. Many of these 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 before. 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. For unrelated parallel machines, we are aware of only three papers by Zhou et al. (2007), Monch (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. One of the prime examples is due to Chen and Powell (1999a). In this paper, 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). The most closely related works to our problem *Rm-TWET* are by Kedad-Sidhoum et al. (2008), Mason et al. (2009), and Al-Khamis and M'Hallah (2011). Kedad-Sidhoum et al. focus on lower bounds for 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. We will elaborate on these further in Section 3 when we introduce our preemptive relaxation. In addition, 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%. 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., Al-Khamis and M'Hallah 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 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 Formulation 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 π_j per unit time if job j completes processing after d_j. Thus, the total weighted tardiness over all jobs is determined as $\sum_j w_j T_j$, 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 ϵ_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.

We first present a time-indexed integer programming formulation for *Rm-TWT* and *Rm-TWET*. Time-indexed formulations were initially introduced for single machine scheduling problems by Dyer and Wolsey (1990). On the one hand, the LP relaxations of these 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 eventually even solving the LP relaxation becomes challenging as we also experience in Section 5. Nevertheless, in the absence of scalable alternate solution approaches for *Rm-TWT* and *Rm-TWET* in the literature, the lower and upper bounds retrieved from the formulation presented below prove useful for measuring the quality of our lower bounds and feasible solutions in small to moderate size instances of *Rm-TWT* and *Rm-TWET*. The time-indexed formulation (**TI**) for *Rm-TWT* and *Rm-TWET* is given next, where the binary variable z_{jkt} takes on the value 1 if job j starts its processing at time t on machine k. The first set of constraints (2) prescribe that each job is started exactly once in the planning horizon on one of the m available machines. The machine capacity constraints (3) ensure that no more than one job is in process at any time instant on any machine.

$$\textbf{(TI)} \qquad \text{minimize} \quad \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{t=0}^{H-p_{jk}} c_{jkt} z_{jkt} \tag{1}$$

$$\text{subject to} \quad \sum_{k=1}^{m}\sum_{t=0}^{H-p_{jk}} z_{jkt} = 1, \qquad j = 1,\ldots,n, \tag{2}$$

$$\sum_{j=1}^{n}\sum_{l=(t-p_{jk}+1)^+}^{t} z_{jkl} \le 1, \qquad k = 1,\ldots,m,\ t = 0,\ldots,H-1, \tag{3}$$

$$z_{jkt} \in \{0,1\}, \qquad j = 1,\ldots,n,\ k = 1,\ldots,m,\ t = 0,\ldots,H-p_{jk}, \tag{4}$$

The notation $(w)^+$ stands for $\max(0,\ w)$, and the objective function coefficients are defined as

$$c_{jkt} = \epsilon_j (d_j - (t + p_{jk}))^+ + \pi_j ((t + p_{jk}) - d_j)^+ \tag{5}$$

without a loss of generality. For *Rm-TWT*, we set $\epsilon_j = 0$ for all jobs j.

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

$$H = \begin{cases} \left\lceil \sum_{j=1}^{n} \max_k \left(p_{jk}\right)/m \right\rceil + p_{\max} & \text{for } Rm\text{-}TWT, \text{ and} \\ \left\lceil \sum_{j=1}^{n} \max_k \left(p_{jk}\right)/m \right\rceil + p_{\max} + d_{\max} & \text{for } Rm\text{-}TWET. \end{cases} \tag{6}$$

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 (6). 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 (6) in the case of a single machine.

3.1 Preemptive Relaxation Shim and Kim (2007a) attack the unweighted version of *Rm-TWT*, the problem $Rm//\sum_j T_j$. As we detail in Section 2, their B&B algorithm does not scale beyond 5 machines and 20 jobs. In their concluding remarks, the authors recognize that the development of stronger lower bounds would be a major step forward toward solving problems of larger size and state "..., 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, and in this section we first briefly review the existing lower bounding methods for parallel machine scheduling problems with additive tardiness and earliness/tardiness objective functions and then 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.

Baptiste et al. (2008) provide a taxonomy of the lower bounding techniques for identical parallel machine scheduling problems with regular objective functions and release dates. We generally agree and follow suit with their classification. A very common and simple set of lower bounds relies on determining a lower bound for the jth 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. For the problem $Pm//\sum T_j$, Azizoglu and Kirca (1998) denote the jth largest processing time and the jth largest due date by $p_{[j]}$ and $d_{[j]}$, respectively. Then, we have $\sum_{i=1}^{j} \frac{p_{[i]}}{m} \leq C_{[j]}$, and $\sum_{j=1}^{n} \max\left(0, \sum_{i=1}^{j} \frac{p_{[i]}}{m} - d_{[j]}\right)$ is a lower bound on the optimal objective value of the original problem. The validity of the expression $\sum_{i=1}^{j} \frac{p_{[i]}}{m} \leq C_{[j]}$ is established by recognizing that the term on the left hand side is the makespan required to carry out the first j shortest jobs by allowing a job to be split and processed simultaneously on all machines. A more sophisticated version of this idea is devised for $Rm//\sum w_j T_j$ by Liaw et al. (2003). The completion times are bounded from below by using similar concepts; however, they are matched to the jobs by solving an assignment problem. Lower bounding techniques based on such *minimal completion times* are developed or employed in several other papers with tardiness related objectives (Koulamas, 1997, 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.

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 these problems solved by column generation yield a prominent class of tight lower bounds. To the best of our knowledge, so far this approach has only been successfully applied to common due date / common due window earliness/tardiness problems (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. for $Pm//\sum w_j T_j$. Kedad-Sidhoum et al. (2008) and Tanaka and Araki (2008) apply Lagrangian relaxation to the time-indexed formulations of their respective identical parallel machine scheduling problems $Pm/r_j/\sum_j \pi_j T_j + \epsilon_j E_j$ and $Pm//\sum T_j$ with successful numerical results (see Section 2). 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 (Baptiste et al., 2008, Kedad-Sidhoum et al., 2008). 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 (**TI**) in Section 5. Kedad-Sidhoum et al. experiment with various relaxations of the problem $Pm/r_j/\sum_j \pi_j T_j + \epsilon_j E_j$ and 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 employ the same Lagrangian relaxation as well. We cite two good reasons for not following a similar path. First, the machine capacity constraints (3) may be aggregated in the identical parallel machine environment, 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 constraints (3) 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 interested reader is referred to Luh et al. (1990), Souayah et al. (2009) for further examples of Lagrangian relaxation applied to TWT problems with identical parallel machines.

Another 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.

$$(\mathbf{TR}) \qquad \text{minimize} \quad \sum_{j=1}^{n}\sum_{t=1}^{H} c'_{jt}x_{jt} \tag{7}$$

$$\text{subject to} \quad \sum_{t=1}^{H} x_{jt} = p_j, \qquad j = 1,\dots,n, \tag{8}$$

$$\sum_{j=1}^{n} x_{jt} \le m, \qquad t = 1,\dots,H, \tag{9}$$

$$0 \le x_{jt} \le 1, \qquad j = 1,\dots,n,\ t = 1,\dots,H. \tag{10}$$

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 (8) mandate that each job j receives p_j units of processing. To observe the machine capacities, constraints (9) 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

$$\sum_{s=t-p_j+1}^{t} c'_{js} \le \epsilon_j (d_j - t)^+ + \pi_j (t - d_j)^+ \quad j = 1,\dots,n,\ t = p_j,\dots,H. \tag{11}$$

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 (11) 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, (11) 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.

$$(\mathbf{TR-A}) \qquad \text{minimize} \quad \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{H} c'_{jkt}x_{jkt} \tag{12}$$

$$\text{subject to} \quad \sum_{t=1}^{H} x_{jkt} = p_{jk}y_{jk}, \qquad j=1,\ldots,n,\ k=1,\ldots,m, \tag{13}$$

$$\sum_{j=1}^{n} x_{jkt} \le 1, \qquad k=1,\ldots,m,\ t=1,\ldots,H, \tag{14}$$

$$\sum_{k=1}^{m} y_{jk} = 1, \qquad j=1,\ldots,n, \tag{15}$$

$$x_{jkt} \ge 0, \qquad j=1,\ldots,n,\ k=1,\ldots,m,\ t=1,\ldots,H, \tag{16}$$

$$y_{jk} \in \{0,1\}, \qquad j=1,\ldots,n,\ k=1,\ldots,m. \tag{17}$$

(**TR − A**) differs from (**TR**) in two main aspects. The capacity constraints (14) appear in a disaggregated form, and all unit jobs of job j are performed on the same machine by constraints (13) and the job partitioning constraints (15). 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 (18) 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.

$$c'_{jkt} = \begin{cases} \frac{\epsilon_j}{p_{jk}}\left[(d_j - \frac{p_{jk}}{2}) - (t - \frac{1}{2})\right] & \text{for } t \le d_j, \text{ and} \\ \frac{\pi_j}{p_{jk}}\left[(t - \frac{1}{2}) - (d_j - \frac{p_{jk}}{2})\right] & \text{for } t > d_j. \end{cases} \tag{18}$$

We next provide a proof that the optimal solution of (**TR − A**) with the cost coefficients given above provides a lower bound on the optimal objective function value of (**TI**). The result is a corollary of Bülbül et al. (2007, Theorem 2.2), where the authors show that the cost coefficients in (18) satisfy (11). Now, 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**($\overline{y}$)) stands for problem (**P**) in which the jobs are assigned to the machines a priori, but the individual machine schedules for this job partition $\overline{y}$ are to be optimized. An optimal schedule is denoted by an asterisk in the superscript.

PROPOSITION 3.1 *The optimal objective function value of* (**TR** − **A**) *with the cost coefficients given by equation* (18), $TC\left(S^*_{\mathbf{TR-A}}\right)$, *is a lower bound on the optimal objective function value* $TC\left(S^*_{\mathbf{TI}}\right)$ *of* (**TI**).

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

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

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

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 (19) 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 (19).

$$-p_{\max} \leq \sum_{j=1}^{n} p_{jk} y_{jk} - \sum_{j=1}^{n} p_{jl} y_{jl} \leq p_{\max}, \qquad k = 1, \ldots, m-1,\ l = k+1, \ldots, m. \tag{19}$$

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 (15) are satisfied, then the model (**TR** − **A**) decomposes into independent transportation problems. We exploit this key observation to design an algorithm based on Benders decomposition for solving (**TR** − **A**) efficiently. Define the set of jobs to be processed on machine k as $J_k = \{j \mid y_{jk} = 1\}$ for $k = 1, \ldots, m$, and set the last period of processing on machine k as appropriate based on (6). Then, we can reformulate (**TR** − **A**) for a fixed $\overline{y}$ as below, where the dual variables associated with the constraints (21) and (22) are indicated in parentheses to the left of their respective constraints:

$$z(\overline{y}) = \text{minimize} \quad \sum_{k=1}^{m} \left(\sum_{j \in J_k} \sum_{t=1}^{H_k} c'_{jkt} x_{jkt} \right) \tag{20}$$

$$(u_{jk}) \qquad \text{subject to} \quad \sum_{t=1}^{H_k} x_{jkt} = p_{jk}\overline{y}_{jk}, \qquad j \in J_k,\ k = 1,\ldots,m, \tag{21}$$

$$(v_{kt}) \qquad \sum_{j\in J_k} x_{jkt} \le 1, \qquad k = 1,\ldots,m,\ t = 1,\ldots,H_k, \tag{22}$$

$$x_{jkt} \ge 0, \qquad j \in J_k,\ k = 1,\ldots,m,\ t = 1,\ldots,H_k. \tag{23}$$

The dual of the model (20)-(23) is then formulated as:

$$z(\overline{y}) = \text{maximize} \quad \sum_{k=1}^{m}\sum_{j\in J_k} p_{jk}\overline{y}_{jk}u_{jk} + \sum_{k=1}^{m}\sum_{t=1}^{H_k} v_{kt} \tag{24}$$

$$\text{subject to} \quad u_{jk} + v_{kt} \le c'_{jkt}, \qquad j \in J_k,\ k = 1,\ldots,m,\ t = 1,\ldots,H_k, \tag{25}$$

$$v_{kt} \le 0, \qquad k = 1,\ldots,m,\ t = 1,\ldots,H_k. \tag{26}$$

Consequently, we obtain the following restricted Benders master problem (**RMP** − **S**), where the current number of cuts is denoted by C, and the optimal values of the dual variables $\overline{u}_{jk}, j \in J_k, k = 1,\ldots,m$, and $\overline{v}_{kt}, k = 1,\ldots,m, t = 1,\ldots,H_k$, in the cut generation subproblem for cut $c, c = 1,\ldots,C$, are represented by $\overline{u}^c_{jk}$ and $\overline{v}^c_{kt}$, respectively. The auxiliary variable η indicates the objective function value of (**RMP** − **S**) and is a lower bound on the optimal objective values of (**TR** − **A**) and (**TI**):

$$(\mathbf{RMP}-\mathbf{S}) \qquad \text{minimize} \quad \eta \tag{27}$$

$$\text{subject to} \quad \sum_{k=1}^{m} y_{jk} = 1 \qquad j = 1,\ldots,n, \tag{28}$$

$$\eta \ge \sum_{k=1}^{m}\sum_{j\in J_k} p_{jk}\overline{u}^c_{jk}y_{jk} + \sum_{k=1}^{m}\sum_{t=1}^{H_k}\overline{v}^c_{kt} \qquad c = 1,\ldots,C, \tag{29}$$

$$y_{jk} \in \{0,1\}, \qquad j = 1,\ldots,n,\ k = 1,\ldots,m. \tag{30}$$

Note that (20)-(23) is feasible and (24)-(26) is bounded for any $\overline{y}$ that satisfies constraints (15). Therefore, no feasibility cuts are required, and only optimality cuts are generated and added iteratively to (**RMP** − **S**) during the course of the algorithm. Furthermore, the cut generation subproblem (20)-(23) decomposes by machine as alluded to at the beginning of this section:

$$z(\overline{y}) = \sum_{k=1}^{m} z_k(\overline{y}), \tag{31}$$

where

$$(\mathbf{TR}_k) \qquad z_k(\overline{y}) = \text{minimize} \quad \sum_{j\in J_k}\sum_{t=1}^{H_k} c'_{jkt}x_{jkt} \tag{32}$$

$$(u_{jk}) \qquad \text{subject to} \quad \sum_{t=1}^{H_k} x_{jkt} = p_{jk}\overline{y}_{jk}, \qquad j \in J_k, \tag{33}$$

$$(v_{kt}) \qquad \sum_{j\in J_k} x_{jkt} \le 1, \qquad t = 1,\ldots,H_k, \tag{34}$$

$$x_{jkt} \ge 0, \qquad j \in J_k,\ t = 1,\ldots,H_k. \tag{35}$$

Thus, we solve m independent single machine transportation problems for generating a *single cut* of the form given in (29). Alternatively, we can obtain a stronger *multi-cut* version of the restricted master problem, (**RMP** − **M**), by defining $\eta_k, k = 1,\ldots,m$, and disaggregating the cuts we generate at the expense of more computational effort for solving the restricted master problem at each iteration:

$$(\mathbf{RMP}-\mathbf{M}) \qquad \text{minimize} \quad \sum_{k=1}^{m}\eta_k \tag{36}$$

$$\text{subject to} \quad (28), (30), \tag{37}$$

$$\eta_k \geq \sum_{j \in J_k} p_{jk} \overline{u}^c_{jk} y_{jk} + \sum_{t=1}^{H_k} \overline{v}^c_{kt} \qquad k = 1, \ldots, m,\ c = 1, \ldots, C. \tag{38}$$

In our preliminary testing, the cut generation algorithm based on (**RMP − S**) was clearly inferior to that based on the multi-cut version (**RMP − M**) in terms of speed, and we only relied on the latter formulation in our computational study in Section 5. Thus, the rest of the paper is exclusively focused on (**RMP − M**). The pseudo-code of the cut generation procedure is stated in Algorithm 1 at the end of the next section following the discussion of our cut strengthening method.

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 of the cut generation subproblem – referred to as 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. For our Benders decomposition scheme laid out before, these issues need a closer look. The dual slave problem for machine k is obtained by taking the dual of (**TR**$_k$):

$$(\mathbf{DS}_k) \qquad \text{maximize} \quad \sum_{j \in J_k} p_{jk} \overline{y}_{jk} u_{jk} + \sum_{t=1}^{H_k} v_{kt} \tag{39}$$

$$\text{subject to} \quad u_{jk} + v_{kt} \leq c'_{jkt}, \qquad j \in J_k,\ t = 1, \ldots, H_k, \tag{40}$$

$$v_{kt} \leq 0, \qquad t = 1, \ldots, H_k. \tag{41}$$

For computational efficiency, the cut generation subproblems only include the set of jobs J_k currently assigned to machine k, and the planning horizon H_k is determined accordingly. However, this amounts to solving the dual slave problem over different feasible regions every time and contradicts the basic pillar of Benders decomposition. Therefore, in order to establish the validity of our approach we must show that an optimal solution of (**DS**$_k$) can always be translated to an optimal solution of (**DS**$_k$ − **F**):

$$(\mathbf{DS}_k - \mathbf{F}) \qquad \text{maximize} \quad \sum_{j=1}^{n} p_{jk} \overline{y}_{jk} u_{jk} + \sum_{t=1}^{H} v_{kt} \tag{42}$$

$$\text{subject to} \quad u_{jk} + v_{kt} \leq c'_{jkt}, \qquad j = 1, \ldots, n,\ t = 1, \ldots, H, \tag{43}$$

$$v_{kt} \leq 0, \qquad t = 1, \ldots, H. \tag{44}$$

In the sequel we argue that an optimal solution of (**DS**$_k$) is not always feasible with respect to (43)-(44). That is, the cuts (29) and (38) may be invalid. This issue does only arise for *Rm-TWET*, and the cut strengthening technique discussed next offers a remedy.

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 regarding their validity pointed out in the previous paragraph. 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 decomposition as well as strengthening the Benders cuts is to prove that we can always augment an optimal solution of $(\mathbf{DS_k})$ to obtain a feasible solution of $(\mathbf{DS_k - F})$ with the same objective function value. This would establish that the augmented solution is optimal for $(\mathbf{DS_k - F})$ 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 (38), the benefit is that we can produce a strengthened Benders cut of the form

$$\eta_k \geq \sum_{j=1}^{n} p_{jk} \overline{u}''_{jk} y_{jk} + \sum_{t=1}^{H} \overline{v}''_{kt} \tag{45}$$

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

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

Proof. Assume that an optimal solution $(\overline{\boldsymbol{u}}_k, \overline{\boldsymbol{v}}_k)$ to $(\mathbf{DS_k})$ 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 $(\overline{\boldsymbol{u}}'_k, \overline{\boldsymbol{v}}'_k)$ is feasible with respect to (40)-(41) 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 $(\mathbf{DS_k})$, 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 $\overline{v}^{\max}_k$. □

Assume that we are given an optimal solution $(\overline{\boldsymbol{u}}'_k, \overline{\boldsymbol{v}}'_k)$ of $(\mathbf{DS_k})$ which satisfies the property in Lemma 4.1. Moreover, we can always extend the planning horizon in $(\mathbf{DS_k})$ 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 $(\overline{\boldsymbol{u}}'_k, \overline{\boldsymbol{v}}''_k)$ is available to $(\mathbf{DS_k})$, 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 (38) is strengthened and then prove that this strengthened cut corresponds to an optimal solution of ($\mathbf{DS_k - F}$).

The variables $u_{jk},\ j \notin J_k$, do not appear in ($\mathbf{DS_k}$) and are implicitly assumed to be zero. Consequently, no term appears on the right hand side of a Benders cut (38) for the jobs that are assigned to other machines in the current restricted master solution $\overline{\boldsymbol{y}}$. 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 (38) 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 $(\overline{\boldsymbol{u}}'_k, \overline{\boldsymbol{v}}''_k)$ of ($\mathbf{DS_k}$):

$$\text{maximize} \quad \sum_{j \notin J_k} p_{jk} u_{jk} \tag{46}$$

$$\text{subject to} \quad u_{jk} \leq c'_{jkt} - \overline{v}''_{kt}, \qquad j \notin J_k,\ t = 1, \ldots, H. \tag{47}$$

The constraints (47) are required to establish that the coefficients of the strengthened cut correspond to an optimal solution of ($\mathbf{DS_k - F}$) – see Proposition 4.1. Clearly, (46)-(47) decomposes by job, and the optimal solution is determined as:

$$\overline{u}''_{jk} = \min \left\{ \min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}), \min_{t=H_k+1,\ldots,H} c'_{jkt} \right\}, \qquad j \notin J_k. \tag{48}$$

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,

$$\min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}) \leq \max_{t=1,\ldots,H_k} c'_{jkt} \leq \min_{t=H_k+1,\ldots,H} c'_{jkt}. \tag{49}$$

Consequently, (48) simplifies to

$$\overline{u}''_{jk} = \min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}), \qquad j \notin J_k, \tag{50}$$

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:

$$\overline{u}''_{jk} = \left\{ \begin{array}{ll} \min\left(\min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}), c'_{jkH_{k+1}} \right) & \text{if } H_k \geq d_j \\ \min\left(\min_{t=1,\ldots,H_k} (c'_{jkt} - \overline{v}''_{kt}), c'_{jkd_j} \right) & \text{if } H_k \leq d_j - 1 \end{array} \right\}, \qquad j \notin J_k. \tag{51}$$

Thus, the strengthened cut finally takes the form specified in (45), 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 (50) or (51), 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 ($\mathbf{DS_k - F}$).

Proposition 4.1 *The dual variables* $(\overline{\boldsymbol{u}}''_k, \overline{\boldsymbol{v}}''_k)$, *which produce a strengthened Benders cut* (45), *are optimal with respect to* ($\mathbf{DS_k - F}$).

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 ($\mathbf{DS_k}$) 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 ($\mathbf{DS_k - F}$). The latter inequalities are enforced directly by the constraints (47). 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 (49), $\overline{u}''_{jk} \leq \max_{t'=1,\ldots,H_k} c'_{jkt'}$ 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 ($\mathbf{DS_k - F}$) is bounded from above by that of ($\mathbf{DS_k}$) because all constraints (40)-(41) are present in (43)-(44), $\overline{y}_{jk} = 0$ 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 ($\mathbf{DS_k - F}$) is clearly identical to that associated with the optimal solution $(\overline{\boldsymbol{u}}'_k, \overline{\boldsymbol{v}}'_k)$ in ($\mathbf{DS_k}$). □

The pseudo-code of our Benders decomposition scheme with the cut strengthening feature for solving (**TR** − **A**) is stated in Algorithms 1-2, and Proposition 4.1 proves its correctness. Skipping the cut strengthening step implicitly implies that an optimal solution $(\overline{u}'_k, \overline{v}'_k)$ of $(\mathbf{DS}_k)$ that conforms with Lemma 4.1 is augmented with zeros as necessary. It is a simple matter to show that this augmented solution is feasible with respect to $(\mathbf{DS}_k - \mathbf{F})$ if we are solving an instance of *Rm-TWT* because the cost coefficients c'_{jkt} are non-negative and non-decreasing over time. However, $\overline{u}''_{jk} = 0,\ j \notin J_k$, is not necessarily feasible for $(\mathbf{DS}_k - \mathbf{F})$ for an instance of *Rm-TWET* according to (51) if $H_k \leq d_j - 1$ because $c'_{jkd_j} < 0$ for $p_{jk} > 1$. Consequently, the resulting cut (38) is invalid. Alternatively, producing a strengthened cut of the form (45) out of the augmented solution $(\overline{u}''_k, \overline{v}''_k)$ obtained by employing either (50) or (51) ensures that the dual slave problem is always solved over the same feasible region and the generated Benders cuts are valid. 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.

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

```
   // Initialization
1  Create (RMP − M) with (36), (28), (30). Add the load balancing constraints (19) for Rm-TWT;
2  repeat                                   // To improve the initial objective value of (RMP − M).
3      Construct a feasible assignment ȳ of jobs to m machines by some heuristic.
4      [cuts, z_1(ȳ), ..., z_m(ȳ)] = generate_cuts(ȳ) ;        // cuts is a collection of m cuts.
5      Add cuts to (RMP − M) as lazy constraints;
6  until some termination condition is satisfied;          // We run a simple dispatch rule once.
   // Main loop
7  Invoke CPLEX on (RMP − M);
8  repeat
9      Identify a new candidate incumbent solution ȳ with an objective value of Σ_{k=1}^m η̄_k;
10     accept_candidate = true;
11     [cuts, z_1(ȳ), ..., z_m(ȳ)] = generate_cuts(ȳ) ;        // cuts is a collection of m cuts.
12     for k = 1 to m do
13         if η̄_k < z_k(ȳ) then                  // ȳ violates some of the missing Benders cuts.
14             Add cuts_k to (RMP − M) as a lazy constraint, accept_candidate = false;
15     end
16 until CPLEX determines that the relative optimality gap of the current incumbent is less than some threshold;
17 The best available job partition ȳ* for (TR − A) is retrieved from CPLEX. If desired, the associated preemptive
   machine schedules are obtained by solving (TR_k) with ȳ* for k = 1, ..., m.
```

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 – for (**RMP** − **M**) the expression $\frac{z(\overline{y}) - \sum_{k=1}^{m} \overline{\eta}_k}{\sum_{k=1}^{m} \overline{\eta}_k}$ – is smaller than a prespecified tolerance level, where the current optimal objective $\sum_{k=1}^{m} \overline{\eta}_k$ of (**RMP** − **M**) is a lower bound on that of (**TR** − **A**) and $z(\overline{y})$ is the objective value of a feasible solution of (**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

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 (45) for $k = 1, \ldots, m$.

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

  Solve ($\mathbf{TR}_k$), retrieve $z_k(\overline{y})$ and the dual optimal solution $(\overline{u}_k, \overline{v}_k)$;

  `/* Calculate an alternate dual optimal solution` $(\overline{u}'_k, \overline{v}'_k)$ `for` ($\mathbf{TR}_k$) `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 a dual optimal solution $(\overline{u}''_k, \overline{v}''_k)$ for ($\mathbf{DS}_k - \mathbf{F}$).

  $\overline{v}''_{kt} = \overline{v}'_{kt},\ t = 1, \ldots, H_k$, 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 (50) or (51), respectively, depending on whether we solve an instance of *Rm-TWT* or *Rm-TWET*;

  Generate and add (45) to *cuts*;

end

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 Section 5.2) are tackled successfully by the exact same solution approach. For both problems, the overarching goal of our computational study is to demonstrate that the proposed Benders-type method solves the preemptive relaxation ($\mathbf{TR} - \mathbf{A}$) 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 $j \in \{1, \ldots, n\}$, 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 $k \in \{2, \ldots, m\}$ 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 ϵ_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 $\left\lfloor U\left[\overline{P}\left(1 - \mathrm{TF} - \frac{\mathrm{RDD}}{2}\right)^+, \overline{P}\left(1 - \mathrm{TF} + \frac{\mathrm{RDD}}{2}\right)\right] \right\rfloor$, 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.

There are 12 combinations of the TF, RDD values and for each combination, 5 instances are generated. Therefore, we

Table 1 Instance generation parameters.

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

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 $\epsilon_j = 0$, $j = 1, \ldots, n$, 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**), m threads are constructed in the lazy constraint callback routine to solve (**TR**$_k$), $k = 1, \ldots, m$, 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*. 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**. We also solve (**TI**) via `CPLEX` under the default parameter settings. The objective value of the root relaxation 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 two sections report the results obtained for *Rm-TWT* and *Rm-TWET*, respectively. We first present the optimality gap results followed by the solution time statistics. For ease of perusal, all tables employ a color formatting scheme so that the values of a performance indicator ranging from worse to better are indicated with varying shades of a color, changing from light to dark.

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 four types of percentage gaps in the table, labeled as "(**TR** − **A**) - **BDS**", "LB Quality", "Feasible Sol'n", and "(**TI**)" in Columns 4–15. For each of these performance indicators, 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 gap 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**. To illustrate the coloring scheme noted before, observe that the numbers in these columns are colored with tints of blue ranging from worst to best performance. 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 time limit of 600 seconds for only 22 instances out of a total of 720. The corresponding number for (**TR** − **A**) - **CPX** is 47 with a time limit of 1800 seconds. The average (& median) gaps of (**TR** − **A**) - **BDS** and (**TR** − **A**) - **CPX** for those instances that could not be solved within the specified time limits are 7.22% (& 4.05%) and 74.14% (& 100%), respectively. Therefore, we conclude that the use of our Benders-type method for solving (**TR** − **A**) is well-justified.

The next three columns under "LB Quality" attest to the quality of the lower bound (LB) provided by (**TR** − **A**) - **BDS** for the optimal objective value of *Rm-TWT*. For a given instance, the expression $\frac{(\text{“Best Integer”}-\text{“LB”})}{(\text{“Best Bound”})}$ provides an upper bound on the gap of LB, where **"Best Integer"** and **"Best Bound"** are the objective function values associated with the best feasible solution retrieved from either (**TR** − **A**) - **BDS** 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%.

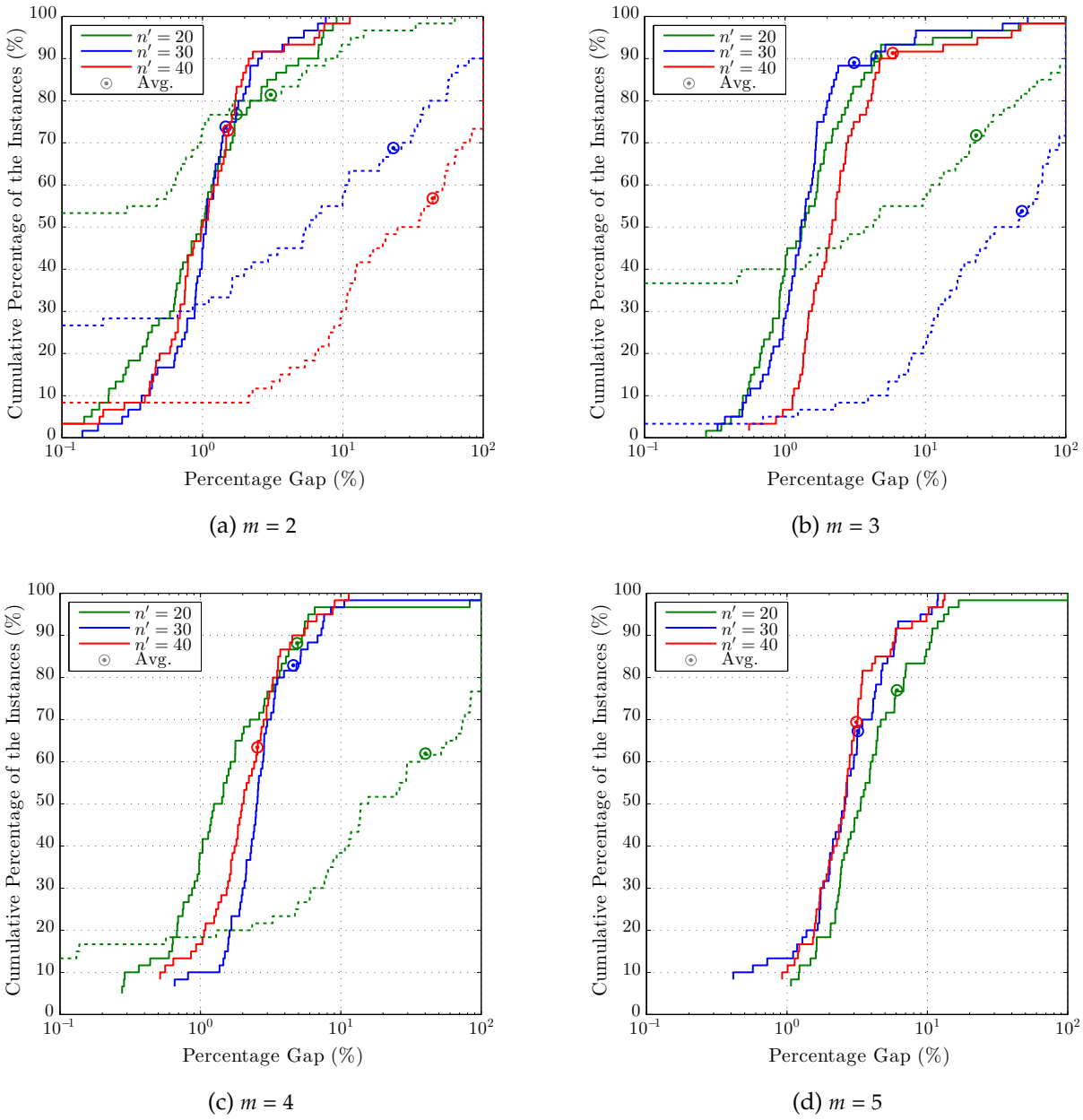
The last six columns under "Feasible Sol'n" and "(**TI**)" present the average upper bounds on the optimality gaps attained by the non-preemptive feasible solutions yielded by (**TR** − **A**) - **BDS** and (**TI**). To facilitate a head-to-head comparison, the values in these six columns are colored together. 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 observe 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.

Table 2 Percentage gap results for *Rm-TWT*.

			(**TR** − **A**) - **BDS**			LB Quality			Feasible Sol'n			(**TI**)		
n	m	TF | RDD	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	0.9	6.1	24.5
		0.6	2.0	2.2	2.2	2.8	3.5	6.4	1.0	1.8	2.2	1.1	1.6	2.4
		0.8	2.5	2.6	2.4	2.8	3.6	3.7	0.8	0.6	0.6	0.0	0.0	0.1
		1.0	2.4	2.1	2.2	2.3	2.4	2.3	0.5	0.5	0.8	0.0	0.0	0.0
60	2	0.4	2.2	1.8	1.6	3.5	5.3	9.5	0.9	2.0	5.0	42.6	54.7	86.1
		0.6	2.4	2.4	2.6	3.3	4.4	5.7	0.9	1.3	1.7	25.6	18.3	28.9
		0.8	2.7	1.7	2.5	3.1	2.7	4.4	1.2	0.6	1.2	4.5	2.7	11.4
		1.0	2.2	2.8	2.8	1.9	2.2	2.5	0.6	1.2	1.0	0.0	0.2	0.0
	3	0.4	2.4	2.1	1.4	5.0	9.3	42.9	1.9	3.6	36.8	61.8	53.5	92.5
		0.6	2.7	2.7	2.8	3.8	5.0	8.7	1.5	1.7	3.0	10.3	24.8	28.7
		0.8	2.8	2.8	2.6	3.1	4.3	4.8	1.4	0.9	0.9	1.5	2.6	0.1
		1.0	2.6	2.5	2.5	2.2	2.6	3.2	0.8	0.6	0.8	0.0	0.0	0.9
80	2	0.4	2.0	2.2	2.0	3.0	5.0	11.6	1.1	2.0	6.8	100.0	96.1	100.0
		0.6	2.2	1.4	2.6	2.9	2.8	4.2	1.0	1.1	0.9	51.8	81.4	45.2
		0.8	2.9	2.3	2.7	3.3	3.0	4.3	1.5	0.7	1.1	14.2	9.5	12.6
		1.0	2.2	2.4	2.3	1.6	2.7	2.7	1.1	0.4	0.6	2.6	6.1	5.8
	4	0.4	2.7	2.6	1.2	6.8	10.4	38.0	3.7	5.0	36.6	89.9	100.0	100.0
		0.6	2.7	2.5	3.8	4.2	5.5	10.1	1.6	2.1	4.2	27.6	48.5	73.8
		0.8	2.8	2.4	2.9	3.6	4.3	5.9	1.1	0.7	1.0	17.1	8.2	11.2
		1.0	2.5	2.8	2.8	2.2	3.1	3.7	0.7	1.1	0.7	0.0	2.0	2.4
90	3	0.4	2.2	2.2	0.9	3.8	7.3	26.9	1.6	3.5	21.2	100.0	100.0	100.0
		0.6	2.6	2.8	2.9	3.3	4.3	6.3	1.2	1.4	2.1	53.6	56.5	73.6
		0.8	2.6	2.5	2.6	3.3	3.8	4.5	1.1	0.9	1.2	14.8	12.0	32.8
		1.0	2.6	2.4	2.4	2.1	3.0	3.2	1.2	0.8	1.0	2.3	9.1	31.2
100	5	0.4	2.7	2.7	2.8	7.3	13.7	20.0	6.2	13.3	20.0			
		0.6	2.7	2.8	5.9	4.2	5.7	12.8	2.9	4.6	9.3			
		0.8	2.6	2.8	3.1	3.7	4.9	6.5	2.3	3.5	4.9			
		1.0	2.5	2.5	2.8	2.8	3.3	4.3	1.4	1.9	2.7			
120	3	0.4	2.5	1.9	1.1	3.7	6.5	42.5	2.3	5.8	42.5			
		0.6	2.2	2.6	2.4	2.9	4.2	5.6	1.6	2.9	4.6			
		0.8	2.6	2.9	2.7	3.0	4.0	4.5	1.5	2.3	2.8			
		1.0	2.8	2.3	2.6	3.0	2.7	3.3	1.1	1.3	1.4			
	4	0.4	2.2	2.9	10.8	4.1	9.1	20.0	3.0	7.9	20.0			
		0.6	2.7	2.6	3.1	3.6	4.4	7.4	2.4	2.8	5.9			
		0.8	2.5	2.5	2.9	3.2	4.1	5.0	1.9	2.6	3.2			
		1.0	2.7	2.7	2.7	3.0	3.2	3.7	1.4	1.6	2.3			
150	5	0.4	2.5	2.9	0.0	5.2	10.0	0.0	4.0	9.0	0.0			
		0.6	2.7	2.7	3.8	3.7	4.6	8.8	2.1	3.3	6.6			
		0.8	2.6	2.7	2.9	3.3	4.3	5.5	1.8	2.8	4.1			
		1.0	1.3	2.5	2.4	1.6	3.1	3.5	0.8	1.7	2.3			
160	4	0.4	2.8	2.8	0.0	4.5	9.2	0.0	3.1	7.9	0.0			
		0.6	2.7	2.9	2.9	3.3	4.2	6.4	1.6	2.7	4.8			
		0.8	2.6	2.2	2.6	3.1	3.3	4.3	1.6	2.2	2.8			
		1.0	2.0	2.0	2.6	2.2	2.4	3.3	0.9	1.1	1.8			
200	5	0.4	2.5	3.4	0.0	4.5	12.3	0.0	3.3	10.8	0.0			
		0.6	2.5	2.8	3.9	3.2	4.5	7.9	1.7	2.7	5.4			
		0.8	2.6	2.7	2.7	3.1	3.9	4.6	1.9	2.2	3.0			
		1.0	2.4	2.0	2.3	2.7	2.4	3.1	2.5	1.8	2.1			

The robustness of the quality of the feasible solutions obtained from our Benders-type approach is further illustrated in Figures 1a–1d. The empirical distributions of the optimality gaps of the feasible solutions associated with (**TR** − **A**) - **BDS** and (**TI**) are plotted with solid and dashed lines in these figures, respectively. The horizontal axes are in logarithmic



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

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

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

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 associated with (**TR** − **A**) - **BDS** 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. In contrast, it is evident that the gaps of the incumbents from (**TI**) degrade quickly with growing values of n'. Half of the instances with $m = 2$ and $n' = 20$ are solved to either optimality or very close to it; however, the median gap becomes 25% as soon as n' is set to 40 with the same number of machines. This effect is persistent for $m = 3$ as well. The ratio of the median gap for $n' = 30$ to that for $n' = 20$ is 8.5. In contrast, for any fixed $m = 2, 3, 4, 5$, the same ratio for any two values of $n' \in \{20, 30, 40\}$ is less than 2 for our algorithm.

Table 3 is similar to Table 2 in structure and format. The average time needed to solve the preemptive relaxation (**TR** − **A**) to within 3% of optimality and the average number of job partitions identified in the search tree by (**TR** − **A**) - **CPX** and (**TR** − **A**) - **BDS** are presented in Columns 4–9. The color formatting is applied to these two sets of columns together to facilitate a direct comparison. The columns labeled as "# of Cuts" report the average number of Benders

Table 3 Average solution times (in seconds) for *Rm-TWT*.

			(TR − A) - CPX*			(TR − A) - BDS*			# of Cuts			Feasible Sol'n			Total			(TI)		
n	m	TF | RDD	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	0.2	0.4	0.6
40	2	0.4	2.3 (2)	3.1 (3)	10.3 (4)	1.6 (9)	1.6 (8)	2.7 (9)	90	83	151	0.08	0.07	0.06	1.7	1.7	2.7	1801	1801	1801
		0.6	4.1 (3)	3.5 (2)	3.7 (2)	1.6 (6)	2.4 (7)	3.1 (9)	71	140	156	0.08	0.08	0.10	1.7	2.5	3.2	1801	1801	1333
		0.8	6.0 (3)	5.2 (3)	4.6 (2)	1.4 (6)	1.0 (5)	3.2 (9)	52	28	154	0.06	0.05	0.09	1.5	1.1	3.3	439	341	472
		1.0	5.8 (2)	5.6 (2)	5.5 (3)	0.6 (3)	1.4 (6)	1.1 (5)	10	47	44	0.03	0.06	0.05	0.7	1.5	1.2	129	103	67
60	2	0.4	8.0 (3)	6.8 (3)	11.5 (3)	4.7 (10)	6.6 (11)	9.7 (11)	89	143	206	0.29	0.30	0.28	5.0	6.9	10.0	1802	1802	1802
		0.6	14.8 (3)	11.5 (3)	11.3 (3)	4.7 (6)	6.9 (9)	17.6 (14)	68	108	331	0.29	0.41	0.50	5.0	7.3	18.1	1808	1802	1805
		0.8	21.5 (3)	18.9 (2)	17.3 (3)	2.8 (5)	6.3 (7)	9.0 (7)	24	96	140	0.19	0.23	0.26	3.0	6.6	9.3	1802	1511	1582
		1.0	24.1 (3)	24.6 (3)	23.6 (3)	1.7 (3)	1.6 (3)	1.7 (3)	10	9	10	0.11	0.09	0.11	1.8	1.6	1.8	1027	858	821
	3	0.4	20.4 (2)	60.1 (4)	215.2 (5)	2.1 (8)	4.0 (13)	3.9 (14)	95	196	210	0.12	0.16	0.11	2.2	4.2	4.0	1802	1803	1804
		0.6	32.3 (2)	29.9 (2)	36.0 (2)	1.1 (5)	2.2 (10)	12.5 (14)	31	70	582	0.09	0.17	0.21	1.2	2.4	12.7	1804	1803	1802
		0.8	39.7 (1)	39.9 (1)	36.4 (2)	0.8 (4)	2.1 (8)	3.2 (8)	18	66	146	0.06	0.11	0.12	0.9	2.2	3.4	1744	1390	956
		1.0	40.7 (1)	44.1 (1)	41.3 (1)	0.8 (4)	0.8 (3)	1.5 (6)	19	16	35	0.05	0.04	0.08	0.9	0.8	1.5	449	673	914
80	2	0.4	14.7 (2)	12.3 (3)	44.3 (4)	8.0 (7)	17.5 (15)	19.6 (13)	66	163	206	0.51	1.11	0.83	8.6	18.6	20.4	1807	1804	1804
		0.6	32.7 (2)	26.7 (3)	30.0 (3)	9.6 (8)	21.1 (13)	35.5 (15)	68	186	316	1.03	1.49	1.66	10.7	22.6	37.2	1804	1804	1804
		0.8	52.3 (3)	56.2 (3)	43.6 (2)	2.7 (3)	8.9 (6)	10.1 (7)	8	51	68	0.30	0.60	0.64	3.0	9.5	10.7	1804	1500	1804
		1.0	68.8 (3)	62.9 (3)	58.8 (2)	2.6 (2)	6.4 (5)	7.0 (6)	8	37	36	0.23	0.42	0.57	2.8	6.9	7.6	1172	1809	1747
	4	0.4	41.6 (2)	294.9 (7)	1716.1 (8)	4.8 (17)	6.6 (18)	11.1 (31)	231	278	310	0.29	0.14	0.16	5.1	6.7	11.3	1804	1805	1804
		0.6	79.0 (2)	72.7 (2)	484.2 (5)	2.0 (8)	7.4 (15)	461.6 (22)	68	421	2149	0.17	0.31	0.40	2.2	7.7	462.0	1811	1807	1805
		0.8	111.3 (1)	103.8 (1)	97.9 (2)	1.3 (5)	5.8 (13)	10.6 (13)	36	283	615	0.10	0.23	0.24	1.4	6.0	10.8	1809	1578	1817
		1.0	118.2 (1)	112.4 (1)	120.8 (1)	1.0 (4)	1.1 (4)	1.7 (7)	22	30	39	0.06	0.07	0.12	1.0	1.2	1.8	1258	1575	1642
90	3	0.4	34.5 (1)	112.2 (3)	1030.3 (7)	5.4 (11)	11.1 (14)	27.5 (26)	95	220	264	0.46	0.54	0.57	5.9	11.6	28.1	1813	1805	1805
		0.6	92.9 (1)	86.9 (1)	93.0 (2)	2.3 (5)	5.0 (8)	24.3 (14)	22	69	479	0.34	0.46	0.72	2.7	5.4	25.1	1805	1805	1805
		0.8	121.8 (1)	121.3 (1)	116.8 (1)	2.0 (4)	2.8 (4)	4.6 (7)	17	26	47	0.24	0.22	0.38	2.3	3.0	5.0	1805	1805	1806
		1.0	147.8 (1)	141.5 (1)	137.7 (1)	1.7 (3)	2.1 (4)	2.1 (4)	13	17	17	0.14	0.18	0.18	1.9	2.3	2.3	1504	1805	1807
100	5	0.4	104.2 (2)	1118.7 (6)	1800.4 (5)	6.4 (15)	13.0 (20)	125.0 (27)	324	429	313	0.31	0.15	0.20	6.7	13.2	125.2			
		0.6	149.5 (1)	167.7 (2)	883.4 (6)	4.4 (10)	15.3 (18)	600.6 (23)	220	718	2951	0.27	0.45	0.50	4.7	15.8	601.1			
		0.8	235.0 (2)	182.4 (1)	241.6 (1)	2.3 (7)	14.7 (21)	308.0 (17)	61	775	2736	0.18	0.46	0.40	2.4	15.1	308.4			
		1.0	253.5 (1)	233.2 (1)	234.9 (1)	1.7 (5)	1.8 (6)	2.7 (7)	44	42	75	0.11	0.12	0.16	1.8	1.9	2.8			

*: Values in the parentheses denote the number of different job partitions present in the final solution pool.

Continued on next page. . .

Table 3 continued...

			(TR − A) - CPX*			(TR − A) - BDS*			# of Cuts			Feasible Sol'n			Total			(TI)		
n	m	TF | RDD	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	0.2	0.4	0.6
120	3	0.4	47.9 (1)	66.2 (1)	1366.7 (4)	7.1 (8)	19.8 (16)	108.6 (25)	42	196	461	0.92	1.48	1.00	8.0	21.3	109.6			
		0.6	205.0 (1)	171.8 (1)	162.3 (2)	4.0 (4)	8.0 (7)	49.0 (14)	19	41	493	0.80	1.22	1.87	4.8	9.2	50.8			
		0.8	262.4 (1)	254.4 (1)	220.8 (1)	3.9 (4)	4.7 (4)	8.2 (7)	16	20	40	0.52	0.59	0.89	4.4	5.3	9.1			
		1.0	337.9 (1)	344.4 (1)	302.9 (1)	3.4 (3)	3.9 (3)	5.7 (5)	13	14	23	0.38	0.49	0.68	3.8	4.4	6.3			
	4	0.4	96.4 (1)	516.2 (5)	1815.9 (4)	12.8 (15)	18.0 (16)	135.3 (31)	258	365	410	0.81	0.74	0.79	13.6	18.8	136.1			
		0.6	205.2 (1)	209.0 (2)	327.0 (3)	3.5 (7)	18.4 (15)	224.9 (21)	42	387	1634	0.53	1.13	1.45	4.1	19.6	226.4			
		0.8	315.6 (1)	314.7 (1)	286.6 (2)	3.4 (5)	5.6 (8)	16.1 (11)	34	62	265	0.41	0.53	0.75	3.9	6.1	16.8			
		1.0	346.2 (1)	325.7 (1)	291.4 (1)	2.1 (3)	2.6 (4)	2.7 (4)	19	25	30	0.17	0.24	0.25	2.3	2.8	3.0			
150	5	0.4	216.0 (2)	1356.5 (6)	1526.8 (3)	19.3 (16)	34.8 (20)	17.8 (23)	410	539	327	1.09	0.74	0.56	20.4	35.6	18.3			
		0.6	379.5 (1)	378.6 (2)	804.2 (4)	5.2 (8)	38.6 (18)	415.3 (24)	58	757	2241	0.81	1.55	1.88	6.0	40.1	417.2			
		0.8	683.5 (1)	645.7 (1)	600.3 (2)	4.2 (5)	10.9 (10)	48.5 (15)	42	154	874	0.50	0.79	1.21	4.7	11.7	49.7			
		1.0	752.4 (1)	713.5 (1)	653.2 (1)	4.8 (6)	3.6 (5)	4.2 (6)	47	35	42	0.47	0.37	0.45	5.3	4.0	4.6			
160	4	0.4	143.3 (1)	451.5 (3)	1361.4 (3)	9.0 (8)	40.2 (15)	28.9 (23)	70	426	255	1.08	1.81	1.11	10.1	42.0	30.0			
		0.6	429.0 (1)	302.3 (1)	704.3 (3)	6.2 (6)	10.7 (9)	175.5 (18)	30	66	1096	1.33	1.82	3.05	7.5	12.5	178.6			
		0.8	713.4 (1)	742.5 (1)	668.1 (1)	5.2 (4)	8.9 (7)	12.3 (8)	24	44	68	0.87	1.20	1.61	6.0	10.1	13.9			
		1.0	934.5 (1)	836.4 (1)	851.1 (1)	5.2 (4)	6.3 (5)	5.7 (4)	23	29	26	0.73	0.88	0.82	5.9	7.2	6.5			
200	5	0.4	345.2 (2)	1476.3 (3)	792.7 (1)	23.9 (12)	324.1 (28)	35.1 (27)	246	1101	268	1.86	3.61	1.41	25.8	327.7	36.6			
		0.6	964.1 (1)	751.5 (1)	1326.9 (3)	10.7 (8)	32.6 (13)	509.7 (26)	61	304	2704	2.21	3.09	5.90	12.9	35.7	515.6			
		0.8	1720.3 (1)	1708.0 (1)	1439.7 (2)	6.7 (5)	29.0 (11)	35.6 (13)	29	233	262	1.16	2.41	2.81	7.9	31.4	38.4			
		1.0	1803.5 (0)	1771.3 (0)	1718.7 (1)	6.9 (4)	8.9 (5)	9.1 (5)	31	43	47	0.92	1.14	1.21	7.9	10.0	10.3			

*: Values in the parentheses denote the number of different job partitions present in the final solution pool.

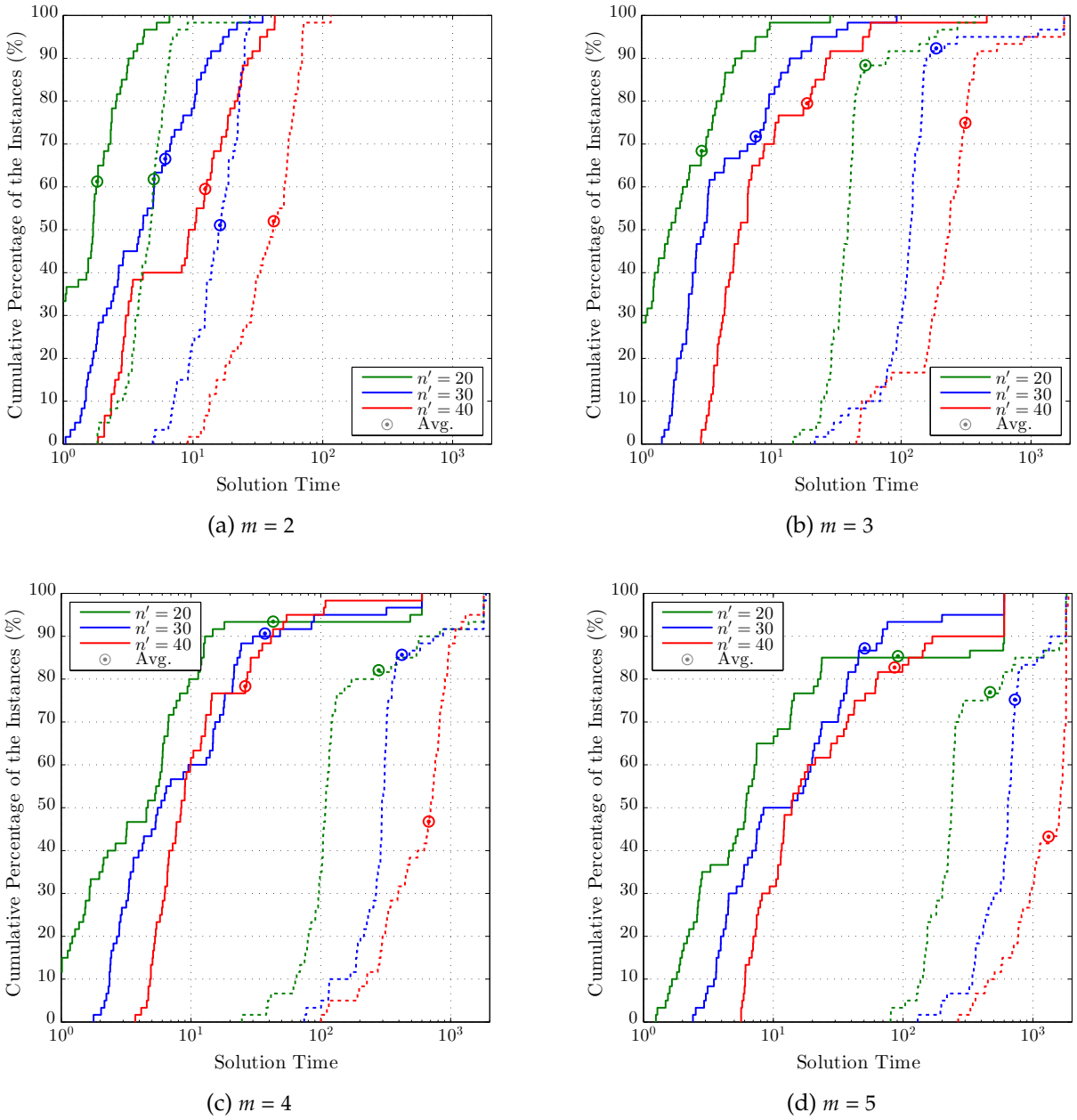
cuts generated in the course of solving (**TR** − **A**) by (**TR** − **A**) - **BDS**. The next three columns, labeled as "Feasible Sol'n", provide the average total time elapsed in the SiPS/SiPSi solver to solve the single machine scheduling problems for all job partitions available at the termination of (**TR** − **A**) - **BDS**. Note that the total number of the single machine scheduling problems solved for an instance is obtained by multiplying the number of job partitions by m. The final six columns under "Total" and "(**TI**)" denote the average total time required by the proposed Benders-type method until the best non-preemptive solution is determined and by CPLEX to solve the time-indexed formulation (**TI**), respectively. For the latter, we do not report any results for the instances with greater than 90 jobs because CPLEX cannot solve the LP relaxation of (**TI**) or find any feasible solution within the time limit of 1,800 seconds in this case. Coupled with its previously demonstrated ability to construct high-quality lower and upper bounds for the original problem, the outstanding total solution time performance of our approach – as evident from the last six columns of Table 3 – makes it a viable alternative for tackling very large instances of *Rm-TWT* successfully.

The solution time performance of (**TR** − **A**) - **BDS** is overwhelmingly superior to that of (**TR** − **A**) - **CPX**. Based on the instances that are solved by both methods within the time limit, the ratio of the solution time of (**TR** − **A**) - **CPX** to that of (**TR** − **A**) - **BDS** is 46.7 on average. Out of a total of 720 instances, only 35 of them take slightly more time to solve for (**TR** − **A**) - **BDS** compared to (**TR** − **A**) - **CPX**. For both methods, instances with loose average due dates within a relatively wide range are more problematic. However, tightening the due dates does also hurt the performance of (**TR** − **A**) - **CPX** while it benefits that of (**TR** − **A**) - **BDS**. The empirical distributions of the solution times of (**TR** − **A**) - **BDS** and (**TR** − **A**) - **CPX** are plotted with solid and dashed lines in Figure 2, 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%.

The columns under "Feasible Sol'n" in Table 3 justify the use of an optimal algorithm to solve the single machine problems for a given job partition. 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**.

5.2 Results for *Rm-TWET* Table 4 is structured identically to Table 2 and depicts the percentage gap 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 the previous section. 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**



(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*.

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.

(**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 in the last six columns of 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 the previous section.

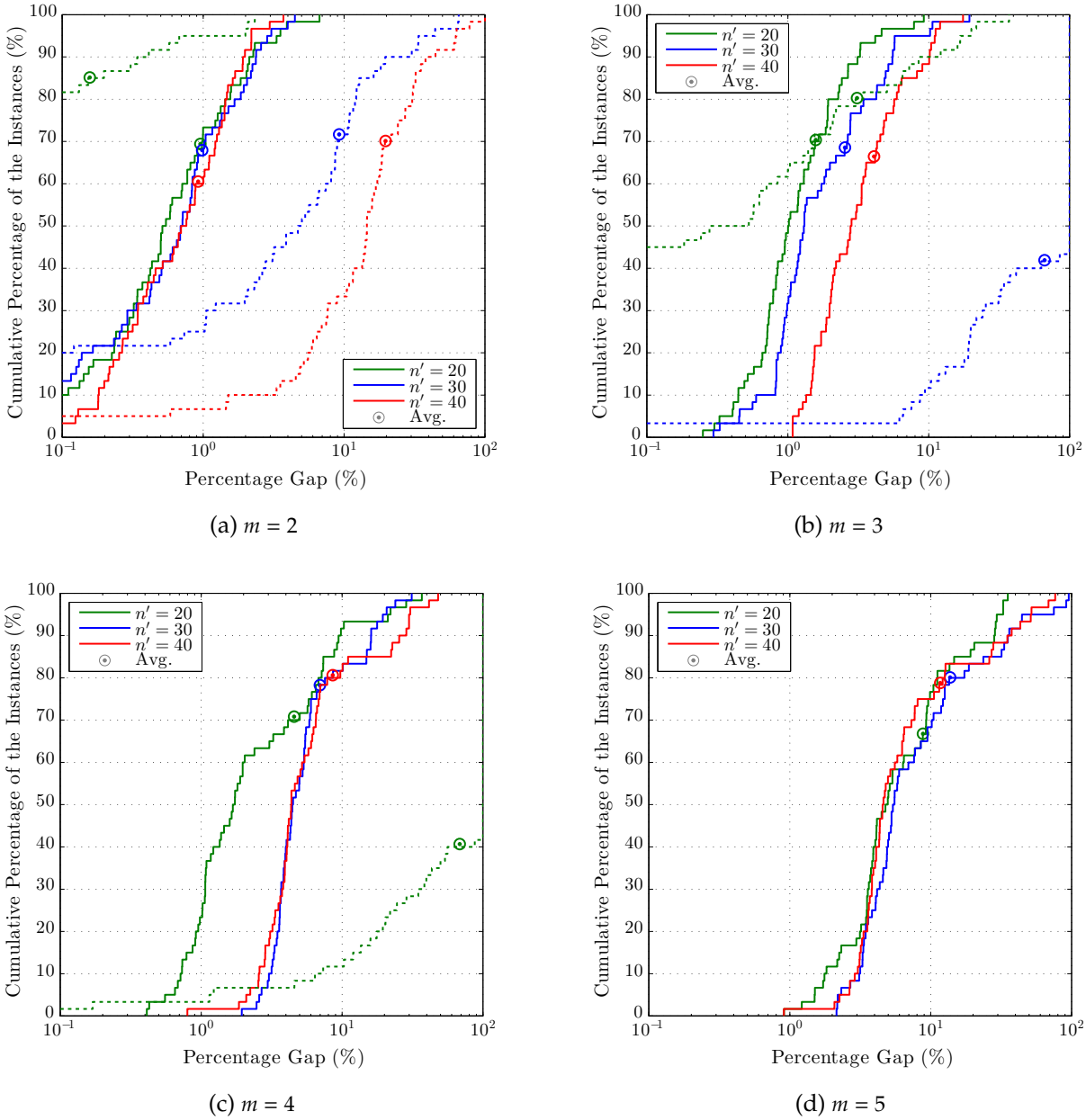
Table 4 Percentage gap results for *Rm-TWET*.

			(**TR − A**) - **BDS**			LB Quality			Feasible Sol'n			(**TI**)		
n	m	TF | RDD	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	0.1	0.1	0.5
		0.6	2.6	2.8	2.9	4.7	7.4	10.3	0.8	2.0	0.7	0.1	1.1	0.0
		0.8	2.0	2.3	2.8	3.1	4.6	5.7	1.0	0.8	0.6	0.0	0.0	0.0
		1.0	1.9	1.4	1.5	2.2	2.2	2.7	0.3	0.2	0.2	0.0	0.0	0.0
60	2	0.4	2.1	3.0	5.4	3.8	6.6	13.7	0.7	1.1	2.7	4.5	3.0	15.7
		0.6	2.2	2.7	3.0	3.9	6.8	10.0	0.7	1.7	2.6	11.4	17.9	19.0
		0.8	1.8	1.8	2.6	2.9	3.8	5.7	0.4	0.6	0.8	7.5	8.3	17.9
		1.0	1.6	0.6	1.7	1.8	1.1	2.8	0.2	0.1	0.3	0.8	1.0	4.0
	3	0.4	2.6	3.0	10.1	5.3	8.1	17.4	1.5	1.3	2.3	2.0	1.5	0.6
		0.6	2.9	2.9	6.0	5.0	7.9	17.9	1.4	2.3	5.3	4.8	3.5	16.3
		0.8	2.0	2.6	2.9	3.4	5.7	7.0	0.9	0.8	1.1	0.4	4.5	3.1
		1.0	2.4	2.4	2.6	2.4	2.7	4.3	0.7	0.7	0.8	0.3	0.0	0.3
80	2	0.4	2.1	2.8	6.6	3.5	5.6	13.3	0.9	1.1	2.6	5.5	14.5	53.2
		0.6	2.0	2.7	3.1	3.1	5.4	7.9	0.5	1.5	1.7	16.9	16.4	32.7
		0.8	1.9	1.7	2.6	2.6	3.1	5.0	0.4	0.5	1.0	33.1	17.3	25.0
		1.0	1.9	1.0	1.0	2.0	1.5	1.9	0.3	0.2	0.3	1.8	8.5	11.9
	4	0.4	2.9	5.9	31.2	5.8	12.1	50.7	2.8	4.9	20.6	81.3	100.0	100.0
		0.6	2.7	3.6	20.3	6.4	9.7	35.6	2.8	4.6	11.5	71.4	72.6	79.5
		0.8	2.4	2.8	3.0	4.2	6.2	8.2	1.1	1.9	1.5	35.4	81.5	49.5
		1.0	2.8	2.1	2.7	3.1	3.1	4.6	1.0	0.9	1.4	23.9	44.3	82.4
90	3	0.4	2.7	3.7	15.2	4.6	8.3	27.3	1.9	3.3	8.6	50.2	92.6	84.0
		0.6	2.1	2.8	6.6	4.0	6.8	15.7	1.2	3.0	5.3	69.7	83.8	97.1
		0.8	2.8	2.7	2.6	3.9	5.0	5.9	0.8	1.6	1.9	33.9	52.2	74.1
		1.0	2.7	2.4	2.2	2.7	3.4	3.4	1.0	0.9	1.0	23.4	49.5	86.9
100	5	0.4	3.1	9.6	34.6	6.1	16.9	57.9	4.2	10.5	30.9			
		0.6	2.9	6.5	22.2	6.1	14.1	41.5	4.5	10.0	22.9			
		0.8	2.6	3.0	6.2	4.7	6.5	12.0	3.4	5.1	7.4			
		1.0	2.5	1.9	2.8	3.0	3.1	5.1	1.4	2.1	3.5			
120	3	0.4	2.4	3.0	11.5	3.8	6.6	20.9	2.5	5.0	12.0			
		0.6	1.9	2.9	5.0	3.2	6.5	12.8	2.0	4.9	9.2			
		0.8	2.7	2.2	2.8	3.6	4.1	5.5	1.5	2.9	3.9			
		1.0	2.7	2.4	2.2	2.9	3.0	3.2	1.3	2.0	1.8			
	4	0.4	2.9	4.3	24.4	5.1	8.6	41.4	3.8	5.9	22.6			
		0.6	2.6	3.5	12.2	4.6	8.6	24.8	4.3	6.5	15.6			
		0.8	2.4	2.8	3.2	3.8	5.4	7.4	3.8	4.8	6.3			
		1.0	2.7	2.5	2.3	3.2	3.4	3.9	3.0	3.4	3.6			
150	5	0.4	2.9	8.3	30.4	4.9	14.8	66.9	4.9	14.8	66.0			
		0.6	2.9	4.8	17.3	5.2	10.8	36.0	5.2	10.8	36.0			
		0.8	2.4	2.9	3.8	4.0	5.7	8.3	4.0	5.7	8.3			
		1.0	2.5	1.9	2.1	3.0	2.8	3.6	3.0	2.8	3.6			

Continued on next page. . .

Table 4 continued...

			(**TR** − **A**) - **BDS**			LB Quality			Feasible Sol'n			(**TI**)		
n	m	TF | RDD	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
160	4	0.4	2.6	4.3	18.1	3.9	8.5	35.1	3.9	8.5	35.1			
		0.6	2.1	3.0	10.6	3.6	6.6	22.7	3.6	6.6	22.7			
		0.8	2.6	2.6	2.8	3.7	4.6	5.8	3.7	4.6	5.8			
		1.0	2.1	2.6	1.8	2.5	3.3	3.0	2.3	3.3	3.0			
200	5	0.4	2.9	6.9	28.3	4.4	11.9	58.4	4.4	11.9	58.4			
		0.6	2.1	2.9	14.9	3.5	7.2	30.7	3.5	7.2	30.7			
		0.8	2.0	2.8	3.0	3.0	4.7	6.4	3.0	4.7	6.4			
		1.0	2.3	2.8	2.7	2.7	3.6	3.9	2.7	3.6	3.9			



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

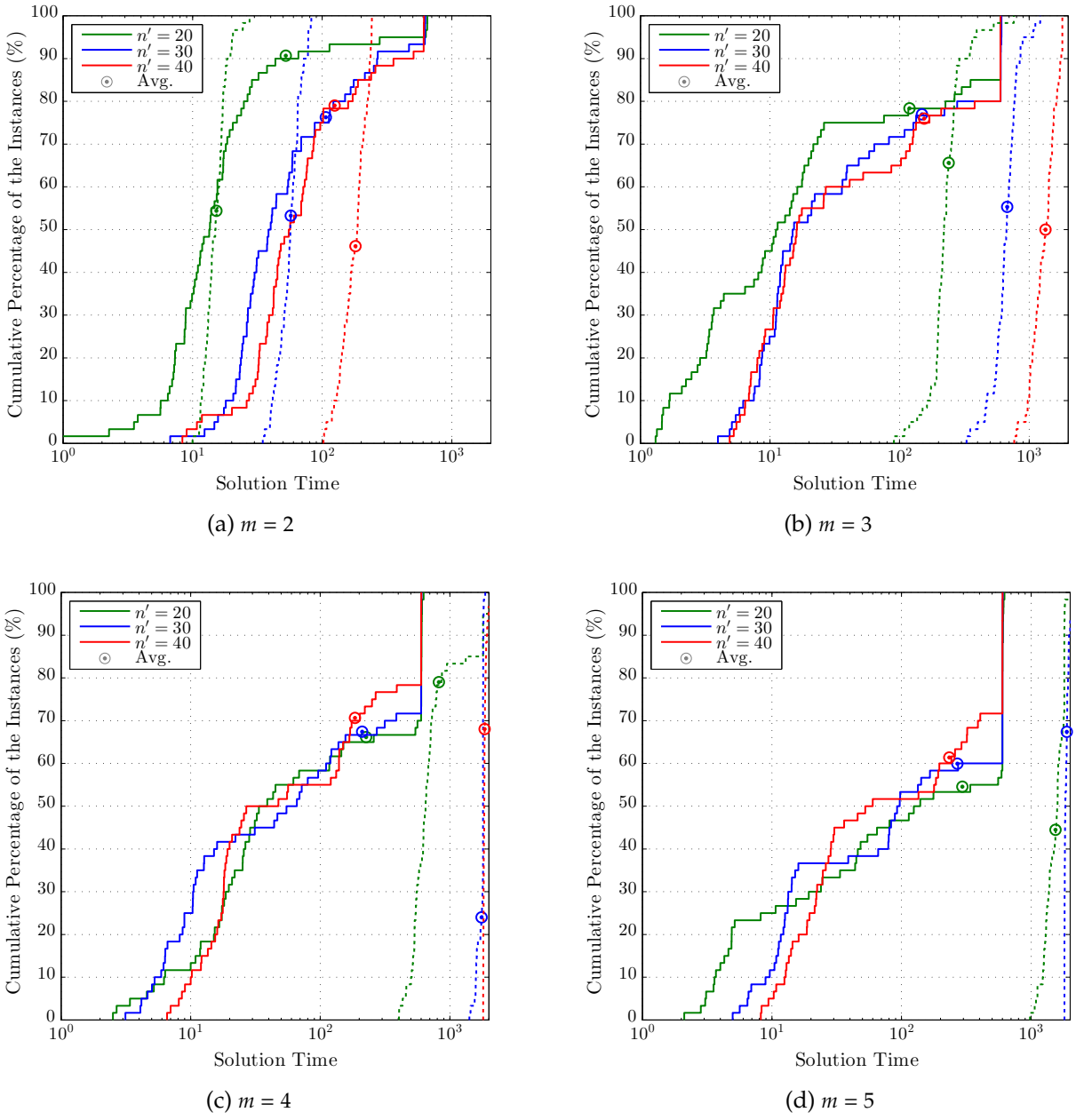
(c) $m = 4$ (d) $m = 5$

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

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** and (**TI**) are indicated with solid and dashed lines in these figures, respectively. As previously, (**TR** − **A**) - **BDS** generally exhibits a robust behavior with respect to varying values of n' for a fixed m. Clearly,

the same cannot be claimed about (**TI**). The problem gets more challenging with an increasing number of machines, and the percentage gaps associated with (**TR** − **A**) - **BDS** demonstrate a modest increase with increasing m. For instance, for 70% of the instances with 2, 3, 4, and 5 machines, the gaps are less than 2%, 5%, 7%, and 11%, respectively. In comparison, the gaps of the incumbents retrieved from (**TI**) grow quickly with m.

The solution times are detailed in Table 5. (**TI**) chokes on instances with greater than 90 jobs, and the corresponding cells are left blank as in Table 3. Moreover, 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 this instance size under "(**TR** − **A**) - **CPX**".



(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*.

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

Table 5 Average solution times (in seconds) for *Rm-TWET*.

			(TR − A) - CPX*			(TR − A) - BDS*			# of Cuts			Feasible Sol'n			Total			(TI)		
n	m	TF | RDD	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	0.2	0.4	0.6
40	2	0.4	15.3 (4)	13.5 (4)	20.9 (6)	15.7 (14)	21.6 (14)	453.7 (18)	485	604	4474	0.8	0.7	1.0	16	22	455	1285	629	1049
		0.6	15.7 (3)	12.1 (3)	13.6 (4)	14.0 (11)	20.7 (12)	48.0 (16)	278	478	1119	0.6	0.6	0.9	15	21	49	1290	1793	682
		0.8	16.9 (3)	14.6 (3)	13.8 (3)	9.6 (11)	9.6 (7)	16.9 (14)	232	243	379	0.6	0.3	0.7	10	10	18	492	652	627
		1.0	15.8 (3)	15.6 (3)	15.2 (3)	4.6 (7)	5.8 (9)	7.8 (10)	132	164	206	0.2	0.3	0.4	5	6	8	253	305	243
60	2	0.4	59.3 (3)	42.6 (3)	47.1 (5)	47.3 (13)	146.1 (23)	550.1 (20)	528	1580	4044	3.0	4.8	5.3	50	151	555	1607	1512	1806
		0.6	62.2 (3)	47.8 (3)	44.6 (4)	41.4 (11)	51.8 (13)	249.3 (21)	312	396	2236	2.6	3.4	5.1	44	55	254	1816	1806	1805
		0.8	64.7 (3)	56.3 (3)	52.9 (3)	24.5 (10)	35.2 (12)	71.8 (14)	203	302	627	2.0	2.4	2.8	26	38	75	1816	1755	1805
		1.0	68.8 (3)	70.9 (3)	69.6 (3)	14.0 (5)	23.0 (10)	26.3 (9)	128	210	225	0.9	1.9	1.8	15	25	28	1371	963	1679
	3	0.4	121.0 (3)	219.3 (4)	384.0 (6)	17.3 (14)	207.3 (17)	601.0 (19)	476	2876	5824	1.2	1.3	1.4	18	209	602	1706	1568	1380
		0.6	198.9 (3)	182.9 (4)	381.7 (6)	6.2 (10)	19.8 (12)	541.2 (18)	136	615	4616	0.8	0.9	1.6	7	21	543	1627	1756	1807
		0.8	256.3 (3)	215.8 (4)	210.8 (4)	2.9 (6)	8.9 (12)	17.8 (15)	48	271	579	0.5	0.8	1.1	3	10	19	1691	1787	1564
		1.0	231.1 (3)	236.5 (3)	231.8 (3)	1.8 (5)	2.2 (6)	5.4 (11)	27	33	133	0.3	0.4	0.8	2	3	6	1259	960	1492
80	2	0.4	198.3 (3)	156.5 (3)	118.2 (4)	68.2 (13)	141.7 (18)	587.9 (22)	366	794	3188	8.0	12.5	14.7	76	154	603	1811	1813	1810
		0.6	193.6 (3)	139.9 (3)	143.2 (4)	47.3 (12)	74.2 (19)	328.0 (26)	294	478	2044	7.2	13.0	18.5	55	87	347	1809	1822	1816
		0.8	210.1 (3)	198.5 (3)	155.5 (3)	35.2 (10)	51.2 (12)	70.1 (13)	256	361	527	5.6	6.4	7.7	41	58	78	1808	1810	1809
		1.0	227.2 (3)	224.3 (3)	203.6 (3)	18.9 (7)	33.5 (11)	39.3 (14)	138	252	283	4.2	7.8	8.2	23	41	47	1240	1809	1809
	4	0.4	432.6 (4)	646.8 (4)	1708.2 (8)	47.9 (16)	596.0 (25)	609.4 (24)	1090	5056	5764	1.6	2.9	3.1	50	599	612	2617	3335	3825
		0.6	603.8 (3)	611.1 (4)	1947.1 (9)	23.4 (17)	324.1 (23)	604.8 (23)	631	2968	6134	2.2	3.5	3.1	26	328	608	2689	3408	3677
		0.8	627.5 (3)	525.9 (3)	644.0 (5)	16.8 (9)	30.7 (15)	426.4 (21)	118	915	3899	1.2	1.7	2.7	18	32	429	2441	2132	3061
		1.0	772.6 (3)	685.2 (3)	680.3 (3)	10.0 (6)	7.7 (7)	17.8 (10)	36	65	110	0.7	0.8	1.4	11	9	19	3237	2625	3288
90	3	0.4	438.6 (3)	495.1 (4)	920.3 (7)	15.7 (9)	335.0 (21)	606.2 (20)	90	2507	4400	3.2	7.1	7.3	19	342	614	2410	3802	3120
		0.6	720.8 (3)	594.9 (4)	757.0 (5)	13.4 (7)	120.4 (18)	605.1 (18)	48	1240	4777	2.8	6.3	6.5	16	127	612	2531	2045	3052
		0.8	800.3 (3)	715.8 (3)	578.1 (3)	9.8 (7)	10.7 (8)	42.8 (13)	39	46	356	2.2	2.9	4.9	12	14	48	1944	2685	2643
		1.0	739.3 (3)	691.0 (3)	669.2 (3)	8.1 (4)	9.8 (6)	10.7 (7)	20	30	41	1.7	1.9	2.4	10	12	13	1908	2444	2551
100	5	0.4	1555.3 (4)	1805.2 (4)	1849.1 (5)	394.2 (25)	610.8 (22)	606.0 (23)	3545	4827	5430	3.6	3.7	3.3	398	614	609			
		0.6	1498.4 (4)	1670.1 (6)	1804.3 (6)	99.5 (22)	600.6 (23)	600.5 (25)	1230	4890	5651	3.6	3.7	3.1	103	604	604			
		0.8	1363.5 (3)	1111.0 (4)	1399.2 (5)	8.4 (12)	109.1 (23)	489.9 (22)	277	2224	4521	1.3	2.5	2.6	10	112	493			
		1.0	1613.7 (3)	1480.1 (3)	1381.9 (3)	3.2 (8)	3.8 (9)	15.9 (14)	66	88	619	0.7	0.9	1.5	4	5	17			

*: Values in the parentheses denote the number of different job partitions present in the final solution pool.

Continued on next page...

Table 5 continued...

			(TR − A) - CPX*			(TR − A) - BDS*			# of Cuts			Feasible Sol'n			Total			(TI)		
n	m	TF | RDD	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	0.2	0.4	0.6
120	3	0.4	1331.1 (3)	1053.2 (4)	977.8 (4)	32.8 (11)	375.7 (19)	600.9 (21)	110	1926	3025	10.6	17.2	22.4	43	393	623			
		0.6	1448.0 (3)	1097.7 (3)	1037.1 (5)	14.6 (8)	135.7 (18)	600.5 (21)	44	869	3472	7.3	17.2	22.1	22	153	623			
		0.8	1678.9 (3)	1416.4 (3)	1103.3 (3)	8.6 (6)	12.7 (7)	54.5 (14)	28	45	406	4.3	6.0	11.1	13	19	66			
		1.0	1699.3 (3)	1726.3 (3)	1480.6 (3)	6.7 (4)	6.6 (4)	10.5 (6)	20	20	38	3.9	3.5	4.8	11	10	15			
	4	0.4	1710.3 (3)	1568.0 (3)	1818.8 (5)	64.3 (14)	600.5 (28)	601.3 (22)	659	4231	4167	5.7	12.1	9.9	70	613	611			
		0.6	1804.6 (2)	1690.9 (4)	1789.4 (5)	18.6 (10)	310.9 (21)	601.5 (21)	170	2736	4461	4.4	10.1	9.8	23	321	611			
		0.8	1812.0 (2)	1801.6 (3)	1692.1 (3)	8.7 (8)	56.3 (15)	244.2 (20)	58	770	2408	3.4	5.6	8.8	12	62	253			
		1.0	1812.1 (2)	1806.3 (2)	1802.9 (2)	4.6 (5)	6.6 (7)	9.5 (9)	32	51	74	1.9	2.4	3.6	6	9	13			
150	5	0.4	1895.2 (1)	1865.2 (2)	2093.3 (2)	154.2 (25)	601.0 (28)	601.6 (25)	1518	4039	3710	11.9	16.1	15.5	166	617	617			
		0.6	1843.1 (2)	1985.6 (2)	1858.6 (2)	46.0 (14)	601.5 (24)	600.6 (24)	495	4237	4459	8.0	14.2	14.0	54	616	615			
		0.8	1858.3 (2)	1857.5 (2)	1849.6 (2)	13.1 (9)	82.8 (18)	507.1 (22)	105	1134	3994	4.7	8.8	11.8	18	92	519			
		1.0	1900.3 (2)	1822.4 (2)	1812.4 (2)	7.0 (6)	10.0 (7)	11.9 (9)	41	83	91	2.8	3.3	4.4	10	13	16			
160	4	0.4	1879.5 (1)	1850.0 (1)	1819.4 (1)	78.2 (13)	488.8 (21)	601.0 (24)	429	2432	2889	14.8	24.4	32.8	93	513	634			
		0.6	1875.9 (1)	1836.1 (1)	1879.6 (1)	23.0 (9)	200.1 (19)	600.6 (27)	77	1314	3391	11.4	23.9	35.5	34	224	636			
		0.8	1906.9 (1)	1841.3 (1)	1843.0 (1)	15.1 (8)	40.6 (11)	137.6 (15)	58	247	1005	8.6	12.0	16.7	24	53	154			
		1.0	1885.6 (1)	1843.6 (1)	1835.8 (1)	10.4 (5)	10.7 (7)	17.9 (10)	34	39	70	6.6	7.9	11.2	17	19	29			
200	5	0.4				126.9 (15)	601.2 (25)	601.3 (20)	568	2709	2609	20.8	37.8	33.8	148	639	635			
		0.6				28.1 (11)	325.3 (23)	601.3 (25)	102	2003	3227	16.9	35.8	42.0	45	361	643			
		0.8				21.6 (10)	80.7 (13)	377.5 (23)	89	567	2467	13.0	17.5	31.7	35	98	409			
		1.0				11.2 (6)	12.6 (5)	19.1 (8)	44	44	89	9.6	7.6	11.1	21	20	30			

*: Values in the parentheses denote the number of different job partitions present in the final solution pool.

90 jobs and generally large RDD values take longer for (**TR** − **A**) - **BDS**. As in Table 3, 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 3, 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 performance under "Feasible Sol'n" in Table 5 is more than satisfactory to solve the single machine problems for a given job partition. The `SiPS/SiPSi` solver returns the optimal solution for a single machine TWET problem in about 27, 110, and 305 milliseconds for instances with $n' = 20, 30, 40$, respectively. These numbers translate into 23 seconds on average to solve all single machine problems to optimality for a five machine and 200 job instance with a maximum of 56 seconds. While these figures are greater than their counterparts for *Rm-TWT*, they still make up for a small part of the total solution time. The time spent for calculating the non-preemptive solutions accounts for only 12.5% of the total solution time on average. Finally, we note that (**TR** − **A**) - **BDS** identifies on average 5.4 times more job partitions per instance compared to (**TR** − **A**) - **CPX**, where the average number of partitions retrieved from the search tree of (**TR** − **A**) - **BDS** is 14.4. As we discussed in the previous section, this is a critical advantage that improves the quality of the best non-preemptive solution.

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.

References

Al-Khamis, T. and M'Hallah, R. (2011). A two-stage stochastic programming model for the parallel machine scheduling problem with machine capacity. *Comput Oper Res*, 38(12):1747–1759.

Armentano, V. c. A. and Yamashita, D. S. (2000). Tabu search for scheduling on identical parallel machines to minimize mean tardiness. *J Intell Manuf*, 11(5):453–460.

Azizoglu, M. and Kirca, O. (1998). Tardiness minimization on parallel machines. *Int J Prod Econ*, 55(2):163–168.

Azizoglu, M. and Kirca, O. (1999a). On the minimization of total weighted flow time with identical and uniform parallel machines. *Eur J Oper Res*, 113(1):91–100.

Azizoglu, M. and Kirca, O. (1999b). Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. *IIE Trans*, 31(2):153–159.

Baker, K. R. and Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: A review. *Oper Res*, 38(1):22–36.

Baptiste, P., Jouglet, A., and Savourey, D. (2008). Lower bounds for parallel machine scheduling problems. *Int J Oper Res*, 3:643–664.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numer Math*, 4(1):238–252.

Biskup, D., Herrmann, J., and Gupta, J. N. (2008). Scheduling identical parallel machines to minimize total tardiness. *Int J Prod Econ*, 115(1):134–142.

Bülbül, K., Kaminsky, P., and Yano, C. (2007). Preemption in single machine earliness/tardiness scheduling. *J Sched*, 10(4-5):271–292.

Chen, Z.-L. and Lee, C.-Y. (2002). Parallel machine scheduling with a common due window. *Eur J Oper Res*, 136(3):512–527.

Chen, Z.-L. and Powell, W. B. (1999a). A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. *Eur J Oper Res*, 116(1):220–232.

Chen, Z.-L. and Powell, W. B. (1999b). Solving Parallel Machine Scheduling Problems by Column Generation. *INFORMS J Comput*, 11(1):78–94.

Cheng, T. and Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *Eur J Oper Res*, 47(3):271 – 292.

Detienne, B., Dauzère-Pérès, S., and Yugma, C. (2011). Scheduling jobs on parallel machines to minimize a regular step total cost function. *J Sched*, 14:523–538.

Dyer, M. and Wolsey, L. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Appl Math*, 26(2–3):255–270.

Fischetti, M., Salvagnin, D., and Zanette, A. (2010). A note on the selection of Benders' cuts. *Math Program*, 124(1-2):175–182.

Graham, R., Lawler, E., Lenstra, J., and Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E. J. and Korte, B., editors, *Discrete Optimization II*, volume 5 of *Ann Discrete Math*, pages 287 – 326. Elsevier.

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, A. and Savourey, D. (2011). Dominance rules for the parallel machine total weighted tardiness scheduling problem with release dates. *Comput Oper Res*, 38(9):1259–1266.

Kanet, J. J. and Sridharan, V. (2000). Scheduling with inserted idle time: Problem taxonomy and literature review. *Oper Res*, 48(1):99–110.

Kedad-Sidhoum, S., Solis, Y. R., and Sourd, F. (2008). Lower bounds for the earliness—tardiness scheduling problem on parallel machines with distinct due dates. *Eur J Oper Res*, 189(3):1305–1316.

Koulamas, C. (1997). Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem. *Nav Res Log*, 44(1):109–125.

Lauff, V. and Werner, F. (2004). Scheduling with common due date, earliness and tardiness penalties for multimachine problems: A survey. *Math Comput Model*, 40(5):637–655.

Lenstra, J., Rinnooy Kan, A., and Brucker, P. (1977). Complexity of machine scheduling problems. *Ann Discrete Math*, 1:343–362.

Liaw, C.-F., Lin, Y.-K., Cheng, C.-Y., and Chen, M. (2003). Scheduling unrelated parallel machines to minimize total weighted tardiness. *Comput Oper Res*, 30(12):1777–1789.

Lin, Y., Pfund, M., and Fowler, J. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Comput Oper Res*, 38(6):901–916.

Luh, P. B., Hoitomt, D. J., Max, E., and Pattipati, K. R. (1990). Schedule generation and reconfiguration for parallel machines. *IEEE T*

Robotic Autom, 6(6):687–696.

Magnanti, T. L. and Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Oper Res*, 29(3):464–484.

Mason, S. J., Jin, S., and Jampani, J. (2009). A moving block heuristic to minimise earliness and tardiness costs on parallel machines. *Int J Prod Res*, 47(19):5377–5390.

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

Pan, Y. and Shi, L. (2007). On the equivalence of the max-min transportation lower bound and the time-indexed lower bound for single-machine scheduling problems. *Math Program*, 110(3):543–559.

Pessoa, A., Uchoa, E., Aragão, M. P., and Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Math Program Comput*, 2(3-4):259–290.

Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer, 3rd edition.

Plateau, M.-C. and Rios-Solis, Y. (2010). Optimal solutions for unrelated parallel machines scheduling problems using convex quadratic reformulations. *Eur J Oper Res*, 201(3):729–736.

Rios-Solis, Y. and Sourd, F. (2008). Exponential neighborhood search for a parallel machine scheduling problem. *Comput Oper Res*, 35(5):1697–1712.

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

Şen, H. and Bülbül, K. (2012). A simple, fast, and effective heuristic for the single-machine total weighted tardiness problem. In Demeulemeester, E. and Herroelen, W., editors, *Proceedings of the 13th Inter. Conf. on Project Management and Scheduling (PMS 2012)*, pages 282–286, Leuven, Belgium.

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

Shim, S.-O. and Kim, Y.-D. (2007a). Minimizing total tardiness in an unrelated parallel-machine scheduling problem. *J Oper Res Soc*, 58(3):346–354.

Shim, S.-O. and Kim, Y.-D. (2007b). Scheduling on parallel identical machines to minimize total tardiness. *Eur J Oper Res*, 177(1):135–146.

Souayah, N., Kacem, I., Haouari, M., and Chu, C. (2009). Scheduling on parallel identical machines to minimise the total weighted tardiness. *Inter J Adv Oper Manage*, 1(1):30–69.

Sourd, F. and Kedad-Sidhoum, S. (2003). The one-machine problem with earliness and tardiness penalties. *J Sched*, 6(6):533–549.

Üster, H. and Agrahari, H. (2011). A Benders decomposition approach for a distribution network design problem with consolidation and capacity considerations. *Oper Res Lett*, 39(2):138–143.

Tanaka, S. and Araki, M. (2008). A branch-and-bound algorithm with Lagrangian relaxation to minimize total tardiness on identical parallel machines. *Int J Prod Econ*, 113(1):446–458.

Tanaka, S. and Fujikuma, S. (2012). A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. *J Sched*, 15:347–361.

Tanaka, S., Fujikuma, S., and Araki, M. (2009). An exact algorithm for single-machine scheduling without machine idle time. *J Sched*, 12:575–593.

van den Akker, J. M., Hoogeveen, J. A., and van de Velde, S. L. (1999). Parallel Machine Scheduling by Column Generation. *Oper Res*, 47(6):862–872.

Van Roy, T. J. (1986). A cross decomposition algorithm for capacitated facility location. *Oper Res*, 34(1):145–163.

Wentges, P. (1996). Accelerating benders' decomposition for the capacitated facility location problem. *Math Method Oper Res*, 44(2):267–290.

Yalaoui, F. and Chu, C. (2002). Parallel machine scheduling to minimize total tardiness. *Int J Prod Econ*, 76(3):265–279.

Zhou, H., Li, Z., and Wu, X. (2007). Scheduling Unrelated Parallel Machine to Minimize Total Weighted Tardiness Using Ant Colony Optimization. In *2007 IEEE Inter. Conf. on Automation and Logistics*, pages 132–136, Jinan. IEEE.