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

algopt
algorithms & optimization

March 7, 2016

An Exact Extended Formulation for the Unrelated Parallel Machine Total Weighted Completion Time Problem

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

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

Keywords: unrelated parallel machines; weighted completion time; Benders decomposition; cut strengthening; exact method; preemptive relaxation; transportation problem.

---

1. Introduction The promise of this paper is to deliver a simple and computationally very effective exact algorithm for minimizing the weighted completion time on a bank of unrelated parallel machines. The objective is one of the most frequently studied fundamental scheduling objectives as it tends to minimize the cycle time of the tasks on the shop floor, and the parallel machine environment is of fundamental practical and theoretical significance. On the one hand, many real manufacturing settings are naturally formulated as parallel machine scheduling problems. Examples discussed in the literature span a variety of industries from semiconductor manufacturing (Shim and Kim, 2007; Detienne et al., 2011) to beverage, printing, and pharmaceutical industries (Biskup et al., 2008). On the other hand, parallel machine scheduling is the immediate generalization of single-machine scheduling from a theoretical point of view, and parallel machine scheduling subproblems are the building blocks of decomposition procedures for multi-stage systems (Pinedo, 2008, p.111). Subproblem solution approaches that deliver high-quality solutions in short computational times are essential to the success of these decomposition methods. Our specific interest in unrelated parallel machines is prompted by a simple observation – capacity expansions over time naturally result in production steps performed on a set of unrelated parallel machines as equipment technology evolves. Shim and Kim (2007), for instance, discuss this issue in the context of semiconductor manufacturing. Thus, there is a clear need for good algorithms tailored to the unrelated parallel machine environment. However, the review of the literature reveals that the research on unrelated parallel machines is scarce compared to a rich literature on identical parallel machine scheduling.

Motivated by the practical and theoretical considerations above, our primary objective in this paper is to devise a scalable effective exact method for solving the problem of minimizing the weighted completion time on a bank of unrelated parallel machines – referred to as *Rm-TWCT* in the rest of the paper. In our problem *Rm-TWCT*, a set of n jobs are ready at time zero to be processed on m unrelated parallel machines. Each job is required to receive service from exactly one machine. All machines are available continuously from time zero onward, and a machine can execute at most one job at a time. Without loss of generality, any machine can process any job, and if job j is performed on machine k, then it stays on the machine for an integer duration of p_{jk} time units without interruption – preemption is not allowed. The objective is to minimize $\sum_j w_j C_j$, where w_j and C_j indicate the unit completion time penalty and the completion time associated with job j, respectively. Following the three field notation of Graham et al. (1979) in classifying scheduling problems, *Rm-TWCT* is characterized as $Rm//\sum_j w_j C_j$. The notation *Rm* in the first field stands for a bank of m unrelated machines. Bruno et al. (1974) prove that minimizing the weighted completion time on two identical parallel machines is $\mathcal{NP}$-hard which also renders *Rm-TWCT* $\mathcal{NP}$-hard.

Any algorithm for a parallel machine scheduling problem has two major components: the jobs are assigned to the machines, and an optimal schedule is determined for each of the machines given the objective function and the job-to-machine assignments. In other words, we may think of a parallel machine scheduling problem as a set partitioning problem where computing the cost of a partition requires solving m independent single-machine scheduling problems. Based on the $\mathcal{NP}$−hardness of the classical set partitioning problem – where each assignment decision is explicitly associated with a fixed cost –, we are already aware that determining the optimal job-to-machine assignments is a challenging task. However, for a parallel machine scheduling problem there may be a second layer of difficulty if the underlying single-machine scheduling problem is not polynomially solvable. This – for instance – is the setting for the problems of minimizing the total weighted tardiness (TWT) and total weighted earliness/tardiness (TWET) on m unrelated parallel machines studied in (Şen and Bülbül, 2015), where the corresponding single-machine problems are strongly $\mathcal{NP}$−hard. These two problems are referred to as *Rm-TWT* and *Rm-TWET* in the sequel, respectively. In contrast, in this paper the cost of a given partition is calculated in polynomial time by applying the well-known weighted shortest processing time (WSPT) rule (Smith, 1956) to each of the m machines separately. This really is the key difference that allows us to turn a mixed integer linear programming (MILP) formulation that only yields lower bounds for *Rm-TWT* and *Rm-TWET* into an exact formulation for *Rm-TWCT*.

The work in (Şen and Bülbül, 2015) was motivated by the lack of strong lower bounds for parallel machine scheduling problems with additive objectives (van den Akker et al., 1999) and focused on the due date related objectives TWT and TWET. Given the two-stage structure inherent in parallel machine scheduling discussed above, the key idea in (Şen and Bülbül, 2015) is to replace the non-preemptive scheduling decisions on a machine by a tight preemptive relaxation solved as a linear program (LP) in an extended variable space. This in turn allows the authors to propose a preemptive relaxation for the original problems *Rm-TWT* and *Rm-TWET*, where the preemptive relaxation is formulated as an MILP amenable to a solution approach that relies on Benders decomposition (Benders, 1962). The job-to-machine assignment decisions are kept in the master problem, and the cost of these decisions is approximated by Benders cuts, generated by solving a separate LP for each machine. In this paper, we prove that the same MILP is an exact formulation for *Rm-TWCT*. This in essence requires that the single-machine weighted completion time problem is solved to optimality as an LP – see Corollary 2.1. Moreover, by exploiting the structure of the TWCT objective we demonstrate that the Benders cuts are generated analytically without the need to invoke any LP algorithm.

These results collectively provide us with a scalable and very effective exact algorithm for *Rm-TWCT*. To put it into perspective, we note that the best performing heuristic for *Rm-TWCT* to date (Rodriguez et al., 2013) runs with a time limit of n seconds, where n is the number of jobs, on instances with up to 1000 jobs and 50 machines. None of the previous studies on heuristics, e.g., (Vredeveld and Hurkens, 2002; Lin et al., 2011; Rodriguez et al., 2012), report results with more than 200 jobs. The computational results in Section 4 illustrate that our exact method solves instances with up to 8 machines and 1000 jobs to optimality within less than 12 seconds on average for any combination of n and m in the indicated range. Instances with 16 and 30 machines are more time consuming, but we impose a time limit of 300 seconds, and instances not solved to optimality within the allotted time almost always terminate with incumbents within 1% of optimality.

We conclude the introduction with a review of the related literature to position our work. The early focus of the parallel machine scheduling literature is on the makespan and total (weighted) completion time objectives with an emphasis on the polynomially solvable problems and approximation algorithms for the makespan (Cheng and Sin, 1990). Pinedo (2008) provides an in-depth discussion of the polynomially solvable cases and the associated structural results of interest. More recent surveys on parallel machine scheduling include (Mokotoff, 2001) and (Blazewicz et al., 2007, Chapter 5). A detailed discussion of the parallel machine scheduling literature on additive due date related performance measures is presented by Şen and Bülbül (2015).

The literature on *Rm-TWCT* can be categorized into three streams: (meta-)heuristics, approximation algorithms, and exact approaches. A good overview of the (meta-)heuristics is provided in (Li and Yang, 2009; Rodriguez et al., 2013). Approximation algorithms for *Rm-TWCT* rely on rounding the optimal solution of a linear or convex quadratic programming relaxation of the problem. We refer the interested reader to (Chekuri and Khanna, 2004), where the authors survey the approximation algorithms for minimizing the total weighted completion time in different machine environments, and to (Li and Yang, 2009). The literature on exact methods for TWCT in the parallel machine environment creates the context for our study, and we restrict our attention to these in the sequel. Interestingly, all exact algorithms for TWCT in the parallel machine environment are limited to identical machines up until 1999. These include the branch-and-bound (B&B) algorithms of Elmaghraby and Park (1974), Barnes and Brennan (1977), Sarin et al. (1988), Belouadah and Potts (1994), and the dynamic programming techniques of Lawler and Moore (1969), Lee and Uzsoy (1992). Unsurprisingly, these papers have very limited success in solving instances of any meaningful size – clearly partly due to the lack of powerful computers back then. The first two B&B procedures for the non-identical parallel machine environment are due to Azizoglu and Kirca (1999a) and Azizoglu and Kirca (1999b). In their earlier work, the authors tackle the problems $Pm//\sum w_jC_j$ and $Qm//\sum w_jC_j$, where Pm stands for the identical parallel machine environment and Qm denotes the presence of m uniform parallel machines. In either case, their B&B algorithm does not scale beyond 3 machines and 25 jobs in 10 and 15 minutes of CPU time for $Pm//\sum w_jC_j$ and $Qm//\sum w_jC_j$, respectively. The first exact solution procedure for our problem *Rm-TWCT* appears in (Azizoglu and Kirca, 1999b). The B&B method of the authors incorporates structural dominance properties to exclude unpromising nodes from consideration. The lower bounding mechanism is based on solving an assignment problem and requires calculating a lower bound on the completion time of each job at each position on each machine. The computational results demonstrate that instances larger than 2 machines and 25 jobs or 3 machines and 20 jobs are beyond the approach with a 15 minute time limit. In two more recent B&B implementations, non-identical job release dates are also taken into account. The B&B method of Yalaoui and Chu (2006) for $Pm/r_j/\sum_j C_j$, where r_j in the second field indicates that jobs may have different

release times, handles at most 45 jobs for 5 and 10 machines and 120 jobs for 2 machines only with a 30 minute time limit. Another B&B algorithm is devised by Nessah et al. (2008) for $Pm/r_j/\sum_j w_j C_j$ and solves instances with up to 60 jobs and 5 machines in one hour of CPU time. In assessing these last two studies, the reader should be aware that the presence of release dates does substantially complicate a scheduling problem in general.

From the discussion above, it is evident that the scalability of B&B methods for parallel machine (weighted) completion time problems is highly doubtful. It turns out that compared to custom B&B procedures, exact solution methods that rely on mathematical programming based decomposition techniques are far more promising for parallel machine scheduling problems with additive objective functions. This is true for both the total (weighted) completion time and the due date related performance measures (Şen and Bülbül, 2015). The underlying reason for this phenomenon is rooted in the tight lower bounds attained by good mathematical programming formulations of parallel machine scheduling problems. Two prime examples, developed contemporaneously to the custom B&B algorithms of Azizoglu and Kirca (1999a) and Azizoglu and Kirca (1999b), are due to Chen and Powell (1999) and van den Akker et al. (1999). In both of these papers, a general parallel machine scheduling problem with an additive objective function of the job completion times is formulated as a set partitioning problem with exponentially many variables. Each variable (column) in the formulation corresponds to a feasible machine schedule. The branch-and-price algorithms of both sets of authors apply column generation to the node LPs due to the huge number of feasible machine schedules, and they mainly differ in their branching schemes. The root relaxation often yields an integer optimal solution, and even if this property does not hold for a particular instance, in a vast majority of cases only a few nodes of the search tree need to be explored until the integer optimal solution is identified. Thus, both studies attribute their relative computational success to the quality of the LP relaxation of their set partitioning formulations. Another common trait of both branch-and-price algorithms is that decreasing the number of machines for a fixed number of jobs has a detrimental effect on the computational performance. Chen and Powell (1999) apply their solution method to *Rm-TWCT* among others and report results with up to 100 jobs and 20 machines. Their average CPU time for instances with 100 jobs degrades from 363 seconds with 20 machines to 2051 seconds with 8 machines. van den Akker et al. (1999) report computational experience only with $Pm//\sum_j w_j C_j$, and their results are similar. Aggregated over all three instance classes the authors consider, the average solution time for instances with 100 jobs and 10 machines is 1512 seconds. Various other exact mathematical programming formulations for *Rm-TWCT* exist, and overviews and comparisons of these are provided in (Vredeveld and Hurkens, 2002; Li and Yang, 2009; Unlu and Mason, 2010). From these discussions, we can infer that the variants of the time-indexed formulation – initially introduced in a seminal paper by Dyer and Wolsey (1990) in the single-machine context – stand out amongst the monolithic MILP formulations. However, the best contender in the literature as an exact solution method for *Rm-TWCT* turns out to be solving the convex quadratic integer programming formulation of Skutella (2001):

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

$$\text{subject to} \quad \sum_{k=1}^{m} y_{jk} = 1, \qquad j=1,\ldots,n, \tag{2}$$

$$y_{jk}\in\{0,1\}, \qquad j=1,\ldots,n,\ k=1,\ldots,m. \tag{3}$$

In (**CQ**), setting the value of the binary variable y_{jk} to one implies that job j is to be processed on machine k. Each job is assigned to exactly one machine by the job partitioning constraints (2). The notation $i \prec_k j$ implies

that either $\frac{w_i}{p_{ik}} > \frac{w_j}{p_{jk}}$ or $\frac{w_i}{p_{ik}} = \frac{w_j}{p_{jk}}$ and $i < j$ following the WSPT order on machine k. (**CQ**) relies on the basic observation $C_j = \sum_{k=1}^{m} y_{jk}\left(p_{jk} + \sum_{i<_k j} p_{ik} y_{ik}\right)$ and a convexification of the resulting objective function. Skutella (2001) proposes this formulation as a means of developing an approximation algorithm for *Rm-TWCT* with a performance guarantee of 3/2. Later, Plateau and Rios-Solis (2010) perform an experimental study on this formulation and find out that it attains the best computational results for *Rm-TWCT* to date. We benchmark the performance of our new MILP formulation solved by Benders decomposition against that of (**CQ**) in Section 4.

The review of the related literature reveals that developing a scalable exact algorithm for *Rm-TWCT* is still an open research question – in particular if the ratio of the number of jobs to the number of machines is not small. This observation and the success of the mathematical programming based solution methods for parallel machine scheduling problems detailed above motivates the work in this paper. In this sense we follow suit with Chen and Powell (1999); van den Akker et al. (1999) and apply a decomposition approach to a mathematical programming formulation that is demonstrated to provide tight lower bounds for the TWT and TWET objectives in the unrelated parallel machine environment (Şen and Bülbül, 2015). In the next section, we present our own formulation for *Rm-TWCT* and prove its correctness before introducing the solution algorithm based on Benders decomposition in Section 3. The computational results in Section 4 attest to the efficacy of our approach, and we conclude in Section 5.

2. An Exact Formulation for *Rm-TWCT* The foundation of our exact formulation for *Rm-TWCT* resides in a class of tight lower bounds initially developed for the single- (Sourd and Kedad-Sidhoum, 2003; Bülbül et al., 2007; Pan and Shi, 2007; Şen and Bülbül, 2012) and identical parallel machine (Kedad-Sidhoum et al., 2008) weighted tardiness and weighted earliness/tardiness scheduling problems. The fundamental idea in this body of work is to allow jobs to be preempted at integer points in time and to penalize the completion time of each unit-length job. The problem of determining the best schedule with this preemptive scheme is then formulated as an assignment or a transportation problem, in which a job j with an integer processing time p_j is allocated a total of p_j unit-length intervals in the planning horizon and no machine executes more than a single unit-length job at a time. Building upon and extending this body of work, Şen and Bülbül (2015) propose the preemptive formulation (**TR** − **A**) below for the unrelated parallel machine environment:

$$(\mathbf{TR}-\mathbf{A}) \qquad \text{minimize} \quad \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{H} c_{jkt} x_{jkt} \tag{4}$$

$$\text{subject to} \quad \sum_{t=1}^{H} x_{jkt} = p_{jk} y_{jk}, \qquad j = 1,\ldots,n,\ k = 1,\ldots,m, \tag{5}$$

$$\sum_{j=1}^{n} x_{jkt} \le 1, \qquad k = 1,\ldots,m,\ t = 1,\ldots,H, \tag{6}$$

$$\sum_{k=1}^{m} y_{jk} = 1, \qquad j = 1,\ldots,n, \tag{7}$$

$$x_{jkt} \ge 0, \qquad j = 1,\ldots,n,\ k = 1,\ldots,m,\ t = 1,\ldots,H, \tag{8}$$

$$y_{jk} \in \{0,1\}, \qquad j = 1,\ldots,n,\ k = 1,\ldots,m. \tag{9}$$

In the model (**TR** − **A**), the time period t represents the time interval $(t-1, t]$, and the variable x_{jkt} is set to one at a cost of c_{jkt} if a unit-length job of job j is performed on machine k in time period t. The machine capacity constraints (6) prescribe that no more than one unit-length job is in process in period t on machine k. The constraints (5) ensure that all unit-length jobs of job j are carried out on the same machine because exactly

one of the binary job-to-machine assignment variables y_{jk}, $k = 1, \ldots m$, is set to one due to the job partitioning constraints (7). The end of the planning horizon $H = \left\lceil \sum_{j=1}^{n} \max_k \left(p_{jk}\right)/m \right\rceil + p_{\max}$, where $p_{\max} = \max_{jk}\left(p_{jk}\right)$, is valid because there exists an optimal solution of the non-preemptive problem *Rm-TWCT* so that all jobs are brought to completion at or before H. See (Şen and Bülbül, 2015, p. 139) for the details of determining the appropriate value of H.

The fundamental difference of (**TR** − **A**) compared to the earlier work in the domain of preemptive relaxations discussed above lies in the constraints (5). As discussed in-depth by Şen and Bülbül (2015), these constraints remove the drawback of the preemptive formulation in (Kedad-Sidhoum et al., 2008) stemming from having the unit-length jobs of a given job distributed over multiple machines; however, they also destroy the desirable polyhedral structure, and we no longer have a transportation problem on our hands. It turns out that (**TR** − **A**) is an MILP formulation of pseudo-polynomial size, which grows very quickly with increasing m, n, and long processing times. The key to solving this formulation effectively is to recognize that (**TR** − **A**) decomposes into into m independent transportation problems for any fixed assignment of the jobs to the machines. This observation renders any integrality restrictions on the variables x_{jkt} redundant – see (8) – and suggests a Benders decomposition algorithm (Benders, 1962) for tackling (**TR** − **A**). The existence of powerful LP engines that can solve very large transportation problem instances in very short times and a fast custom procedure to strengthen the Benders cuts enables Şen and Bülbül (2015) to obtain tight lower bounds for *Rm-TWT* and *Rm-TWET* by solving (**TR** − **A**) to (near-)optimality. Our main contribution over (Şen and Bülbül, 2015) in this paper is to prove that (**TR** − **A**) with the objective coefficients to be discussed next yields an exact formulation for TWCT. Moreover, we characterize the analytic closed form of the optimal solutions of the Benders subproblems and generate cuts without the need of invoking an LP solver as was the case in (Şen and Bülbül, 2015). Coupling this with further enhancements attained in the Benders cut strengthening procedure of Şen and Bülbül (2015) by exploiting the special structure of the optimal solutions of the Benders subproblems results in a scalable and very fast optimal solution technique for *Rm-TWCT*.

2.1 Exactness of (TR − A) One issue that deserves special attention is the choice of the objective coefficients in (**TR** − **A**). In the context of the preemptive relaxations developed previously for earliness/tardiness scheduling problems, the particular choice determines the strength of the lower bound and the empirical performance, and the existing papers in the literature differ from each other in this respect. For an in-depth discussion on this subject and the properties of alternate cost coefficients, the reader is referred to (Pan and Shi, 2007; Şen and Bülbül, 2015). In this paper, we directly employ the cost coefficients of Şen and Bülbül (2015) for *Rm-TWT* and *Rm-TWET* by setting all due dates equal to zero – both problems reduce to *Rm-TWCT* with zero due dates:

$$c_{jkt} = \frac{w_j}{p_{jk}}\left(t + \frac{p_{jk}}{2} - \frac{1}{2}\right), \qquad j = 1, \ldots, n,\ k = 1, \ldots, m,\ t = 1, \ldots, H. \tag{10}$$

Consequently, Proposition 2.1 presented next follows as a direct corollary of Şen and Bülbül (2015, Proposition 3.1).

Proposition 2.1 *The optimal objective function value of* (**TR** − **A**) *with the cost coefficients given in* (10) *is a lower bound on the optimal objective function value of the original non-preemptive problem Rm-TWCT.*

In the rest of the paper, any reference to (**TR** − **A**) employs the set of cost coefficients (10). Two further intermediate results proven next and Proposition 2.1 collectively yield our main result formalized in Theorem 2.1. In the sequel, a non-preemptive feasible solution of (**TR** − **A**) refers to a feasible solution of (**TR** − **A**) in which all unit-length jobs of any job are processed in consecutive periods.

LEMMA 2.1 *The cost charged against any non-preemptive feasible solution of* (**TR** − **A**) *is identical to the cost incurred by this schedule in the original non-preemptive problem Rm-TWCT.*

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

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

□

PROPOSITION 2.2 *There exists a non-preemptive optimal solution of* (**TR** − **A**). *Furthermore, in this optimal solution the jobs assigned to each machine are sequenced in the WSPT order.*

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

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

$$(\mathbf{AP}) \qquad \text{minimize} \quad \sum_{i=1}^{P_k} \sum_{t=1}^{P_k} c'_{it}\, \delta_{it} \tag{11}$$

$$\sum_{t=1}^{P_k} \delta_{it} = 1, \qquad i = 1, \ldots, P_k, \tag{12}$$

$$\sum_{i=1}^{P_k} \delta_{it} = 1, \qquad t = 1, \ldots, P_k, \tag{13}$$

$$\delta_{it} \in \{0, 1\}, \qquad i = 1, \ldots, P_k,\ t = 1, \ldots, P_k. \tag{14}$$

The formulation (**AP**) decouples the unit-length jobs of a given job and regards them as independent tasks. The first p_{1k} tasks belong to job 1, the next p_{2k} tasks are associated with job 2, and so on. The original job associated with task i is denoted by $j(i)$. The binary variable δ_{it} takes on the value one at a cost of $c'_{it} = c_{j(i)kt}$ – as defined in (10) – if task i is processed in period t. The constraints (12)-(13) mandate that each task is assigned to one period and vice versa, respectively.

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

$$c'_{i_1 t_1} + c'_{i_2 t_2} \leq c'_{i_1 t_2} + c'_{i_2 t_1}, \qquad 1 \leq i_1 < i_2 \leq P_k,\ 1 \leq t_1 < t_2 \leq P_k. \tag{15}$$

To recognize this, we note that $c'_{it} = c_{j(i)kt} = \frac{w_{j(i)}}{p_{j(i)k}} \left(t + \frac{p_{j(i)k}}{2} - \frac{1}{2}\right)$ and verify (15) for any $i_1 < i_2$ and $t_1 < t_2$:

$$\frac{w_{j(i_1)}}{p_{j(i_1)k}} \left(t_1 + \frac{p_{j(i_1)k}}{2} - \frac{1}{2}\right) + \frac{w_{j(i_2)}}{p_{j(i_2)k}} \left(t_2 + \frac{p_{j(i_2)k}}{2} - \frac{1}{2}\right) \leq \frac{w_{j(i_1)}}{p_{j(i_1)k}} \left(t_2 + \frac{p_{j(i_1)k}}{2} - \frac{1}{2}\right) + \frac{w_{j(i_2)}}{p_{j(i_2)k}} \left(t_1 + \frac{p_{j(i_2)k}}{2} - \frac{1}{2}\right)$$

$$\iff \quad \frac{w_{j(i_1)}}{p_{j(i_1)k}}t_1 + \frac{w_{j(i_2)}}{p_{j(i_2)k}}t_2 \leq \frac{w_{j(i_1)}}{p_{j(i_1)k}}t_2 + \frac{w_{j(i_2)}}{p_{j(i_2)k}}t_1$$

$$\iff \quad \left(\frac{w_{j(i_1)}}{p_{j(i_1)k}} - \frac{w_{j(i_2)}}{p_{j(i_2)k}}\right)t_1 \leq \left(\frac{w_{j(i_1)}}{p_{j(i_1)k}} - \frac{w_{j(i_2)}}{p_{j(i_2)k}}\right)t_2. \tag{16}$$

The final inequality (16) holds because $t_1 < t_2$ and $i_1 < i_2$ implies $\frac{w_{j(i_1)}}{p_{j(i_1)k}} \geq \frac{w_{j(i_2)}}{p_{j(i_2)k}}$.

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

Theorem 2.1 (**TR** − **A**) *is an exact formulation for Rm-TWCT.*

Proof. By Proposition 2.2, we can identify a non-preemptive optimal solution S^* of (**TR** − **A**). The associated objective value is a lower bound on the optimal objective value of *Rm-TWCT* based on Proposition 2.1. Moreover, S^* is also feasible for *Rm-TWCT*, and Lemma 2.1 assures that the cost it incurs with respect to *Rm-TWCT* is identical to the optimal objective value of (**TR** − **A**). Therefore, S^* must be an optimal schedule for *Rm-TWCT*. □

(**TR** − **A**) is thus an exact extended formulation for *Rm-TWCT* obtained through variable splitting because the completion time C_j of job j is the minimum value such that $C_j \geq tx_{jkt},\ k = 1,\ldots,m,\ t = 1,\ldots,H$.

The corollary below follows from Theorem 2.1 and suggests an LP-based alternative for solving the non-preemptive single-machine TWCT problem.

Corollary 2.1 *The non-preemptive single-machine total weighted completion time problem is equivalent to a transportation problem of pseudo-polynomial size.*

Proof. Theorem 2.1 assures that we can solve the single-machine TWCT problem to optimality by setting $m = 1$ in (**TR** − **A**). However, in this case, the formulation is simplified by setting $H = \sum_j p_{j1}$ and dropping the binary variables y_{jk} from the formulation along with the constraints (7). The resulting model is a transportation problem with n source nodes and $\sum_j p_{j1}$ sink nodes, where the objective coefficients are defined by (10). The size of the transportation problem is pseudo-polynomial because the number of sink nodes depends on the magnitude of the processing times. □

The primal solution of the transportation problem entails assigning the values of the variables $x_{j1t},\ j = 1,\ldots,n,\ t = 1,\ldots,H$, as prescribed by the WSPT order of the jobs, and the closed form of the dual solution is specified in the next section as part of our Benders decomposition scheme – see (27). In either case, the solution procedure is very fast in practice; however, there is no way of getting around the theoretical pseudo-polynomial complexity because of the number of value assignments required.

The result in Corollary 2.1 was actually discovered previously in the context of the relaxations of the single-machine TWCT problem with release dates – the problem $1/r_j/\sum_j w_jC_j$. Dyer and Wolsey (1990) explore and compare the strengths of various relaxations of $1/r_j/\sum_j w_jC_j$. Their weaker time-indexed formulation (D) (Dyer and Wolsey, 1990, Section 5) boils down to (**TR** − **A**) with $m = 1$ after some simple manipulation. The authors point out that the optimal objective value of this preemptive time-indexed formulation can be computed in $O(n \log n)$ time based on a lower bounding algorithm proposed in (Posner, 1985) for the single-machine TWCT problem with deadlines. A discussion of the same transportation problem as a relaxation for $1/r_j/\sum_j w_jC_j$ is also presented by Goemans et al. (2002) who design approximation algorithms for this problem. Thus, in a sense Corollary 2.1 is a unifying result. It is obtained by studying a special case

of the preemptive time-indexed formulations of earliness/tardiness scheduling problems and offers a new perspective on an already known result in different contexts. However, from our point of view the primary significance of being able to solve the single-machine TWCT problem as a transportation problem derives from the valuable dual information extracted from the optimal LP solution. In decomposition algorithms for complex scheduling problems with a single-machine TWCT component, one may opt for solving the subproblems as an LP in pseudo-polynomial time for the sake of this dual information – as is the case in this paper. We are not aware of any other paper in the literature which adopts a similar approach.

3. Solving (TR − A) via Benders Decomposition (**TR** − **A**) decomposes into m independent transportation problems for any fixed partition of the jobs to the machines as specified by the values of the binary variables y_{jk}, $j = 1,\dots,n, k = 1,\dots,m$. Thus, for any given fixed $\overline{y}$ satisfying (7), (**TR** − **A**) is reformulated via the Benders decomposition principle by replacing the right hand side of the set of constraints (5) by $p_{jk}\overline{y}_{jk}$ and removing the set of constraints (7) and (9) from the model. The resulting LP is referred to as (**TR** − **A**($\overline{y}$)), and the dual variables associated with the set of constraints (5) and (6) are denoted by u_{jk}, $j = 1,\dots,n$, $k = 1,\dots,m$, and v_{kt}, $k = 1,\dots,m$, $t = 1,\dots,H$, respectively. The formulation below is then the dual of (**TR** − **A**($\overline{y}$)) and exposes the decomposition into m independent transportation problems:

$$z(\overline{y}) = \sum_{k=1}^{m} z_k(\overline{y}), \tag{17}$$

where

$$(\mathbf{DS}_k) \qquad z_k(\overline{y}) = \text{maximize} \quad \sum_{j=1}^{n} p_{jk}\overline{y}_{jk}u_{jk} + \sum_{t=1}^{H} v_{kt} \tag{18}$$

$$\text{subject to} \quad u_{jk} + v_{kt} \le c_{jkt}, \qquad j = 1,\dots,n,\ t = 1,\dots,H, \tag{19}$$

$$v_{kt} \le 0, \qquad t = 1,\dots,H, \tag{20}$$

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

We denote the feasible region (19)-(20) of (**DS**$_k$) by Q_k. Then, for each fixed job partition $\overline{y}$, (**DS**$_k$), $k = 1,\dots,m$, provide extreme point optimal solutions $(\overline{u}_k, \overline{v}_k) \in Q_k$, $k = 1,\dots,m$, so that the sum of the optimal objective function values of (**DS**$_k$), $k = 1,\dots,m$, is equal to the cost of the best solution of (**TR** − **A**) that can be attained from the job partition $\overline{y}$. Consequently, a Benders optimality cut of the form

$$\eta \ge \sum_{k=1}^{m} \left(\sum_{j=1}^{n} p_{jk}\overline{u}_{jk}y_{jk} + \sum_{t=1}^{H} \overline{v}_{kt} \right) \tag{21}$$

removes $\overline{y}$ from further consideration, where η represents a lower bound on the optimal objective value of (**TR** − **A**). Moreover, note that (**TR** − **A**($\overline{y}$)) is always feasible and only optimality cuts need to be generated. In the resulting relaxed Benders master problem (**RMP**) presented below, the cuts (21) appear in the disaggregated form (24) because this so-called *multi-cut* version proved superior in our preliminary computational experiments.

$$(\mathbf{RMP}) \qquad \text{minimize} \quad \sum_{k=1}^{m} \eta_k \tag{22}$$

$$\text{subject to} \quad \sum_{k=1}^{m} y_{jk} = 1, \qquad j = 1,\dots,n, \tag{23}$$

$$\eta_k \geq \sum_{j=1}^{n} p_{jk}\overline{u}^c_{jk} y_{jk} + \sum_{t=1}^{H} \overline{v}^c_{kt}, \qquad k = 1,\ldots,m,\ c = 1,\ldots,C, \tag{24}$$

$$y_{jk} \in \{0,1\}, \qquad j = 1,\ldots,n,\ k = 1,\ldots,m. \tag{25}$$

In (**RMP**), the number of times the dual slave problems ($\mathbf{DS_k}$), $k = 1,\ldots,m$, have been solved so far is designated by C, and the superscript c in $\overline{u}^c_{jk}$ and $\overline{v}^c_{kt}$ indicates that these parameters are the components of the extreme point optimal solution $(\overline{\boldsymbol{u}}^c_k, \overline{\boldsymbol{v}}^c_k) \in Q_k$ of ($\mathbf{DS_k}$) obtained in iteration c of the cut generation. The auxiliary variable η_k approximates the total cost charged against the jobs performed on machine k from below, and the objective function value $\sum_{k=1}^{m} \eta_k$ of (**RMP**) is therefore a lower bound on the optimal objective value of ($\mathbf{TR-A}$). We also remark that *Rm-TWCT* does always possess an optimal solution that fulfills the load balancing constraints (26). Otherwise, the final job on machine k may be appended to the end of the schedule on another machine without increasing the total cost (Azizoglu and Kirca, 1999b, Theorem 1). These m constraints, which are slightly stronger compared to the trivial load balancing constraints $\sum_{j=1}^{n} p_{jk} y_{jk} \leq H,\ k = 1,\ldots,m$, are incorporated into the initial (**RMP**).

$$\sum_{j=1}^{n} p_{jk} y_{jk} \leq \frac{1}{m}\left(\sum_{j} \max_{l}\left\{p_{jl}\right\} + \sum_{l\neq k} \max_{j}\left\{p_{jl}\right\}\right), \qquad k = 1,\ldots,m. \tag{26}$$

The classical textbook application of Benders decomposition iterates between generating Benders cuts based on the optimal solution of the current relaxed master problem and re-optimizing the relaxed master problem with the additional cuts starting from a brand-new search tree. Consequently, the same nodes may be re-visited several times during the course of the Benders decomposition algorithm resulting in an inefficient implementation. With the recent advances in solver technology, a Benders type algorithm may be executed on a single search tree by exploiting the *lazy constraint* feature (IBM ILOG CPLEX, 2012) that allows generating a Benders cut for each candidate incumbent solution. Thus, no integer solution is evaluated more than once, and this generally leads to very substantial computational savings. In-depth discussions are offered in (Rubin, 2011; Şen and Bülbül, 2015). The use of the *lazy constraint callback* routine is also reflected in the pseudo-code of our optimal algorithm for ($\mathbf{TR-A}$) stated in Algorithm 1 in the appendix.

Next, we turn our attention to the efficient generation of the Benders optimality cuts (24) by providing a closed form optimal solution to ($\mathbf{DS_k}$) without having to resort to a generic LP or transportation problem solver as was the case in (Şen and Bülbül, 2015). Note that ($\mathbf{DS_k}$) is defined over the entire set of jobs and the full length of the planning horizon H. However, a job j' with $\overline{y}_{j'k} = 0$ is not performed on machine k and may be excluded from consideration while optimizing ($\mathbf{DS_k}$). Therefore, we define $J_k = \{j \mid \overline{y}_{jk} = 1\}$ as the set of jobs to be processed on machine k and $H_k = \sum_{j\in J_k} p_{jk}$ as the associated planning horizon, respectively, and solve a restricted version of ($\mathbf{DS_k}$) – referred to as ($\mathbf{DS_k - R}$) – over these jobs and time periods only. The optimal solution of ($\mathbf{DS_k - R}$) is then trivially augmented to an optimal solution of ($\mathbf{DS_k}$) by setting $\overline{u}_{jk} = 0$ for $j \notin J_k$ and $\overline{v}_{kt} = 0$ for $t = H_k + 1,\ldots,H$. The validity of this augmentation derives easily from the discussion in (Şen and Bülbül, 2015, Section 4.1). The primal problem associated with ($\mathbf{DS_k - R}$) is the restricted cut generation subproblem ($\mathbf{TR_k - R}$). Based on Corollary 2.1, ($\mathbf{TR_k - R}$) is equivalent to the non-preemptive single-machine TWCT problem solved over the jobs in J_k by scheduling all unit jobs of a given job contiguously by following the WSPT order. In the presentation below, the jobs assigned to machine k are re-labeled in the WSPT order; that is, $i < j$ implies $\frac{w_i}{p_{ik}} \geq \frac{w_j}{p_{jk}}$ for two jobs $i, j \in J_k$. Then, we claim that

$$\begin{aligned} \overline{u}_{jk} &= \frac{w_j}{p_{jk}}\left(\textstyle\sum_{i\leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2}\right) + \sum_{i>j} w_i, && j \in J_k, \\ \overline{v}_{kt} &= \frac{w_{l(t)}}{p_{l(t)k}}\left(t - \textstyle\sum_{i\leq l(t)} p_{ik}\right) - \sum_{i>l(t)} w_i, && t = 1,\ldots,H_k, \end{aligned} \tag{27}$$

is an optimal solution for $(\mathbf{DS_k} - \mathbf{R})$, where $l(t)$ is defined such that $\sum_{i<l(t)} p_{ik} < t \leq \sum_{i\leq l(t)} p_{ik}$ and denotes the job processed on machine k in period t in the optimal solution of $(\mathbf{TR_k} - \mathbf{R})$. Note that attaching intuitive meanings to the values in (27) is relatively simple by taking a sensitivity analysis viewpoint and interpreting $\overline{u}_{jk}$ and $\overline{v}_{kt}$ as the shadow prices associated with the corresponding constraints in (5) and (6), respectively. In particular, assume that the capacity of the "resource" time period t is increased by one unit and two unit-length jobs are carried out simultaneously in this period. Consequently, the unit-length jobs of job $l(t)$ currently processed in periods $t+1,\ldots,\sum_{i\leq l(t)} p_{ik}$ are completed one time unit earlier at a cost savings of $\frac{w_{l(t)}}{p_{l(t)k}}\left(\sum_{i\leq l(t)} p_{ik} - t\right)$. Similarly, all unit-length jobs of the jobs following job $l(t)$ are shifted one period to the left leading to a decrease of $\sum_{i>l(t)} \frac{w_i}{p_{ik}} p_{ik} = \sum_{i>l(t)} w_i$ in the total cost. Putting these two cost savings figures together provides us with the value of $\overline{v}_{kt}$. Assuming that the processing time of job j is reduced by a single time unit and following a similar analysis we can arrive at the value given for $\overline{u}_{jk}$ in (27). To this end, observe that the first term on the right hand side of the expression for $\overline{u}_{jk}$ is the cost incurred by the final unit-length job of job j performed in period $\sum_{i\leq j} p_{ik}$. The optimality of $(\overline{u}_k, \overline{v}_k)$ presented in (27) for $(\mathbf{DS_k} - \mathbf{R})$ is formalized in the next proposition. The proof follows a standard recipe from linear programming duality and is relegated to the appendix.

Proposition 3.1 *$(\overline{u}_k, \overline{v}_k)$ specified in (27) is an optimal solution for $(\mathbf{DS_k} - \mathbf{R})$ with the cost coefficients given in (10).*

The general consensus of the literature (Magnanti and Wong, 1981; Fischetti et al., 2010) is that algorithms based on Benders decomposition rarely deliver a good computational performance unless the Benders cuts are strengthened. The essence of the matter is to choose a "good" optimal solution of the dual slave problem to generate cuts if primal degeneracy is present in the cut generation subproblem. In the context of this study, the transportation problem is renowned for it is primal degeneracy and an optimal solution of $(\mathbf{DS_k})$ obtained by extending the optimal solution $(\overline{u}_k, \overline{v}_k)$ of $(\mathbf{DS_k} - \mathbf{R})$ given in (27) by setting $\overline{u}_{jk} = 0$ for $j \notin J_k$ and $\overline{v}_{kt} = 0$ for $t = H_k + 1,\ldots,H$, results in weak cuts and uncompetitive computational performance. To alleviate this issue, we apply the cut strengthening procedure of Şen and Bülbül (2015) which yields an alternate optimal solution $(\overline{u}'_k, \overline{v}'_k)$ of $(\mathbf{DS_k})$ (Şen and Bülbül, 2015, Proposition 4.1):

$$\begin{array}{llll} \overline{u}'_{jk} = \overline{u}_{jk}, & j \in J_k, & \overline{u}'_{jk} = \min_{t=1,\ldots,H_k}(c_{jkt} - \overline{v}_{kt}), & j \notin J_k, \\ \overline{v}'_{kt} = \overline{v}_{kt}, & t = 1,\ldots,H_k, & \overline{v}'_{kt} = 0, & t = H_{k+1},\ldots,H. \end{array} \tag{28}$$

The benefit is that y_{jk}, $j \notin J_k$, are now added to the right hand side of (24) with strictly positive coefficients $p_{jk}\overline{u}'_{jk}$, $j \notin J_k$. Note that $\overline{u}'_{jk} > 0$ for all $j \notin J_k$ because $c_{jkt} > 0$ in the entire planning horizon for all jobs and $\max_{t=1,\ldots,H_k} \overline{v}_{kt} = \overline{v}_{kH_k} = 0$. The naive calculation of $\overline{u}'_{jk}$ for all $j \notin J_k$ requires $O(nH)$ operations; however, by investigating and exploiting the structure of $(\overline{u}_k, \overline{v}_k)$ we can carry out this calculation in $O(n)$ time based on Lemma 3.1 and the ensuing discussion. Consequently, the pseudo-polynomial complexity $O(mnH)$ for strengthening all m cuts in one iteration of the Benders decomposition algorithm in (Şen and Bülbül, 2015) is reduced to the polynomial complexity $O(mn)$ for *Rm-TWCT* in this paper. This enhancement stems from the following result:

Lemma 3.1 *For a given job $j \notin J_k$, the function $f_{jk}(t) = c_{jkt} - \overline{v}_{kt}$ defined over $t = 1,\ldots,H_k$ is discrete convex.*

Proof. Similar to the convention in the proof of Proposition 3.1, assume that the jobs in J_k are re-labeled in the WSPT order and define $l(t)$ such that $\sum_{i<l(t)} p_{ik} < t \leq \sum_{i\leq l(t)} p_{ik}$. Recall that $l(t)$ is the job processed on machine k in period t in the optimal solution of $(\mathbf{TR_k} - \mathbf{R})$. Furthermore, $\left(t - \sum_{i\leq l(t)} p_{ik}\right) = -(p_{hk} - 1)$ in the first period t assigned to a job $h = l(t)$, and this difference is increased by one in each following period job h

is processed until it becomes zero upon the completion of job h. Consequently, for two consecutive periods t, $t+1$ assigned to job h such that $l(t+1) = l(t) = h$, we have $\overline{v}_{k,t+1} - \overline{v}_{kt} = \frac{w_h}{p_{hk}}$. Otherwise, if job h completes processing in period t and job $h+1$ is started in period $t+1$, then $l(t) = h$, $l(t+1) = h+1$ and we obtain $\overline{v}_{k,t+1} - \overline{v}_{kt} = \left(-\frac{w_{h+1}}{p_{h+1,k}}\left(p_{h+1,k} - 1\right) - \sum_{i>h+1} w_i\right) - \left(-\sum_{i>h} w_i\right) = \frac{w_{h+1}}{p_{h+1,k}}$. We conclude that

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

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

$$\Delta_{jk}(t) = f_{jk}(t+1) - f_{jk}(t) = \left(\frac{w_j}{p_{jk}}(t+1) - \overline{v}_{k,t+1}\right) - \left(\frac{w_j}{p_{jk}}t - \overline{v}_{kt}\right) = \frac{w_j}{p_{jk}} - (\overline{v}_{k,t+1} - \overline{v}_{kt}) = \frac{w_j}{p_{jk}} - \frac{w_{l(t+1)}}{p_{l(t+1)k}} \tag{29}$$

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

The discrete convexity of $f_{jk}(t)$ for $j \notin J_k$ implies that $\overline{u}'_{jk} = \min_{t=1,\ldots,H_k}(c_{jkt} - \overline{v}_{kt}) = c_{jkt^*_{jk}} - \overline{v}_{kt^*_{jk}}$, where $t^*_{jk} = \min\left\{t = 1, \ldots, H_k \mid \Delta_{jk}(t) \geq 0\right\}$ with the understanding that $\Delta_{jk}(H_k) \geq 0$. A further key observation allows us to conduct the search for t^*_{jk} over the set of jobs in J_k instead of the set of time periods $1, \ldots, H_k$. Recall that the optimal solution of $(\mathbf{TR_k} - \mathbf{R})$ is non-preemptive, and $l(t) = h$ from period $\sum_{i<h} p_{ik} + 1$ until period $\sum_{i\leq h} p_{ik}$. Consequently, (29) assures that $\Delta_{jk}(t) = \frac{w_j}{p_{jk}} - \frac{w_h}{p_{hk}}$ from period $C_{h-1} = \sum_{i<h} p_{ik}$ until period $C_{h-1} + p_h - 1$, and the period in which $\Delta_{jk}(t)$ changes sign must coincide with the completion time of a job in J_k – except when $\frac{w_j}{p_{jk}} \geq \max_{i\in J_k} \frac{w_i}{p_{ik}}$ and $t^*_{jk} = 1$. Moreover, from (29) we can also infer that $t^*_{j_1k} \leq t^*_{j_2k}$ is satisfied for two jobs $j_1, j_2 \notin J_k$ so that $\frac{w_{j_1}}{p_{j_1k}} \geq \frac{w_{j_2}}{p_{j_2k}}$. Putting these ideas together, all $\overline{u}'_{jk}$, $j \notin J_k$, can be computed in $O(n)$ time by traversing both these jobs and the jobs in J_k in the WSPT order – see Algorithm 3 in the appendix.

The pseudo-code of our complete Benders decomposition scheme with the cut strengthening feature for solving $(\mathbf{TR} - \mathbf{A})$ is stated in Algorithms 1-3 in Appendix G. The finiteness of the algorithm is argued through the finite number of job partitions.

4. Computational Study The overall goal of our computational study is to demonstrate that the proposed Benders decomposition algorithm – referred to as $(\mathbf{TR} - \mathbf{A})$ - **BDS** in the rest of the paper – has a great computational performance both in absolute and relative terms. We solve instances across a broad range of (n, m) combinations with both short and long processing times and investigate the effectiveness of our algorithm in order to establish its absolute performance. It turns out that $(\mathbf{TR} - \mathbf{A})$ - **BDS** scales very well as instances with up to 1000 jobs and 30 machines are either solved to optimality with a time limit of five minutes or very high-quality incumbents are obtained at termination. For $m \leq 8$, the optimal solution is attained within 10 seconds for a great majority of the instances for any n, and we conclude that $(\mathbf{TR} - \mathbf{A})$ - **BDS** is even fast enough to be employed as a subroutine in decomposition algorithms designed for the more general flexible flow- and job shop scheduling problems. Furthermore, to argue that $(\mathbf{TR} - \mathbf{A})$ - **BDS** is the best exact algorithm for *Rm-TWCT* developed to date, we benchmark it against the convex quadratic integer programming formulation $(\mathbf{CQ})$ presented in Section 1 and solved by an off-the-shelf engine. This approach is referred to as $(\mathbf{CQ})$ - **CPLEX** in the sequel. As pointed out in Section 1, $(\mathbf{CQ})$ - **CPLEX** represents the current state-of-the-art for the exact methods designed for *Rm-TWCT*. The results reveal that compared to $(\mathbf{CQ})$ - **CPLEX**, $(\mathbf{TR} - \mathbf{A})$ - **BDS** either determines the optimal solution in considerably shorter time or it identifies an incumbent of substantially higher quality at the time limit. The details of our analyses are presented in the following.

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

The computational results are obtained on a personal computer with a 2.33 GHz Intel® Core™2 Quad processor Q8200 and 8 GB of memory running on Windows 7. (**TR** − **A**) - **BDS** is implemented in `C++` using the `Concert Technology` component library of `IBM® ILOG® CPLEX® 12.5`. Under the default parameter settings, the implementation of a control callback – such as the lazy constraint callback – leads `CPLEX` to turn off its dynamic search feature and apply a traditional branch-and-cut strategy with a single thread (IBM ILOG CPLEX, 2012). Therefore, to exploit parallelism and promote simultaneous cut generation, `CPLEX` is allowed to use up to four parallel threads – as specified by the `Threads` parameter – with the `ParallelMode` switch set to `Opportunistic`. Moreover, based on the positive previous experience of the authors in (Şen and Bülbül, 2015) the `MIPEmphasis` switch, which "controls the trade-offs between speed, feasibility, optimality, and moving bounds in MIP," takes on the value four in order to emphasize finding high-quality hidden feasible solutions. (**CQ**) - **`CPLEX`** calls `CPLEX` to solve (**CQ**) with the default parameter settings, except that `Threads=4`, `ParallelMode=Opportunistic`, and `MIPEmphasis=4` for a fair comparison with (**TR** − **A**) - **BDS**. In both methods, `CPLEX` terminates the optimization if the relative optimality gap drops below `EpGap`=10^{-3}=`0.1%`, or the working memory exceeds `WorkMem=5,120`=5 GB, or the time expended reaches `TiLim=300` seconds. More details on these parameters are available in (IBM ILOG CPLEX, 2012).

Table 1 consists of 22 rows, one for each possible combination of n and m listed in the first two columns. Each figure in the table represents a statistic over 10 instances. The number of instances solved to optimality within the time limit appears in the columns labeled with "#", and the columns under "%Gap" and "Time" (in seconds) present the average optimality gaps retrieved from `CPLEX` at termination and the average solution times, respectively. Note that `CPLEX` uses the formula $\frac{|best\_bound - best\_integer|}{10^{-10}+|best\_integer|}$ for computing the optimality gap of an instance (IBM ILOG CPLEX, 2012), where *best_bound* is the largest available lower bound and *best_integer* is the objective value of the incumbent at termination. A color formatting scheme is applied separately to each of the three performance measures "%Gap", "Time", and "#," so that the values ranging from better to worse are indicated with colors changing from green towards red. The results for instances with relatively short processing times are reported in the left half of the table in Columns 3-8 under the heading "$p_{\max} = 20$." The remaining columns depict the performance measures for the corresponding instances with $p_{\max} = 100$.

The results in Table 1 underline that (**TR** − **A**) - **BDS** provides provably optimal solutions for the majority of the instances well within the time limit of 300 seconds. More specifically, (**TR** − **A**) - **BDS** solves 343 out of a total of 440 instances to optimality in 3.73 seconds on average with a maximum solution time of 162.48 seconds. In contrast, (**CQ**) - **`CPLEX`** attains only 270 optimal solutions in 13.95 seconds on average with a maximum of 241.51 seconds. The average and maximum gaps of (**TR** − **A**) - **BDS** for those 97 instances that

Table 1 Average optimality gap and solution time (in seconds) results for *Rm-TWCT*.

		$p_{\max} = 20$						$p_{\max} = 100$					
		(TR − A) - BDS			**(CQ) - CPLEX**			**(TR − A) - BDS**			**(CQ) - CPLEX**		
n	m	%Gap	Time	#	%Gap	Time	#	%Gap	Time	#	%Gap	Time	#
30	2	0.00	0.06	10	0.00	0.05	10	0.00	0.06	10	0.00	0.04	10
	4	0.00	0.11	10	0.00	0.15	10	0.00	0.14	10	0.00	0.14	10
	6	0.00	0.22	10	0.00	0.54	10	0.00	0.30	10	0.00	0.71	10
	8	0.00	0.60	10	0.00	3.91	10	0.00	0.67	10	0.00	1.06	10
100	2	0.00	0.13	10	0.00	0.14	10	0.00	0.13	10	0.00	0.11	10
	4	0.00	1.00	10	0.00	1.60	10	0.00	1.07	10	0.00	1.78	10
	6	0.00	1.62	10	0.26	70.08	8	0.00	1.61	10	0.40	123.54	6
	8	0.00	15.21	10	0.71	153.28	5	0.00	9.15	10	1.13	219.34	3
	16	0.67	273.19	1	5.91	182.04	4	0.84	260.67	2	7.30	300.00	0
	30	2.12	300.00	0	16.13	300.00	0	3.80	300.00	0	30.49	300.00	0
400	2	0.00	0.05	10	0.00	1.16	10	0.00	0.05	10	0.00	1.47	10
	4	0.00	0.56	10	0.00	3.28	10	0.00	0.41	10	0.00	5.79	10
	6	0.00	0.82	10	0.22	240.67	2	0.00	0.70	10	0.16	191.02	4
	8	0.00	1.03	10	0.30	151.50	5	0.00	1.72	10	0.28	239.19	3
	16	0.13	300.00	0	1.86	248.95	3	0.22	300.00	0	1.86	300.00	0
	30	0.39	300.00	0	3.75	300.00	0	0.66	300.00	0	6.00	300.00	0
1000	2	0.00	0.10	10	0.00	13.22	10	0.00	0.10	10	0.00	19.21	10
	4	0.00	0.87	10	0.00	27.43	10	0.00	0.79	10	0.00	34.66	10
	6	0.00	1.72	10	0.09	109.50	8	0.00	1.46	10	0.00	49.49	10
	8	0.00	2.67	10	0.14	247.60	2	0.00	1.97	10	0.11	203.59	4
	16	0.00	51.00	10	0.83	257.90	2	0.00	5.93	10	0.63	281.08	1
	30	0.30	300.00	0	-	-	-	0.17	300.00	0	-	-	-

could not be solved to optimality within the specified time limit are just 0.96% and 5.51%, respectively. The corresponding figures for **(CQ) - CPLEX** are 5.21% and 75.45% over 150 instances. **(CQ) - CPLEX** cannot handle the remaining 20 largest instances with $n = 1000$, $m = 30$ and terminates due to an out-of-memory error. The differences between **(TR − A) - BDS** and **(CQ) - CPLEX** become more apparent if we separate out the groups of instances solved to optimality by both methods and those not solved to optimality by either method within the time limit. On 198 of the 263 instances in the earlier group, **(TR − A) - BDS** outpaces **(CQ) - CPLEX** by an average (& maximum) factor of 32.48 (& 228.63) computed from the ratios of the solution times of **(CQ) - CPLEX** to those of **(TR − A) - BDS**. On six instances the solution times are identical, and on the remaining 59 instances **(CQ) - CPLEX** is on average 2.44 times faster, where the corresponding maximum is 10.90. In the second group of 70 instances, **(TR − A) - BDS** attains a smaller optimality gap at termination for 67 instances. The difference in the optimality gaps is on average 9.37% and reaches a maximum of 70.93%. **(CQ) - CPLEX** yields a smaller terminal gap on just three instances and the difference does not exceed 1.81%. In addition,

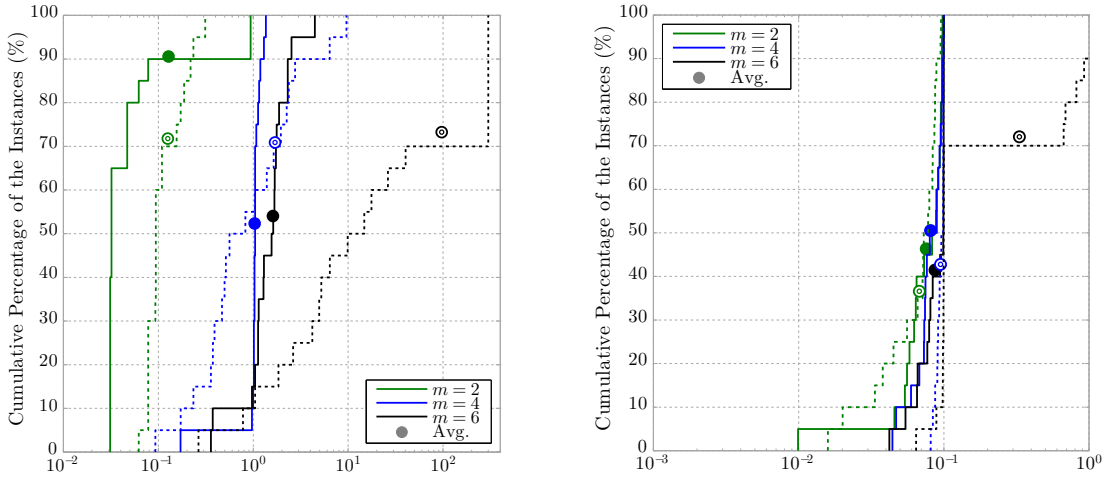
note that there are only 7 instances for which (**TR − A**) - **BDS** is only able to provide an incumbent at the time limit while (**CQ**) - **CPLEX** solves these instances optimally. In comparison, (**TR − A**) - **BDS** supplies optimal solutions for 80 instances that remain unsolved at the time limit by (**CQ**) - **CPLEX** and obtains incumbents very close to optimality with an average gap of 0.24% for the 20 instances with $n = 1000$, $m = 30$, while these instances are completely beyond the reach of (**CQ**) - **CPLEX** due to insufficient memory. To conclude, we stress that (**TR − A**) - **BDS** is clearly the exact algorithm of choice for *Rm-TWCT* because it either delivers an optimal solution substantially faster or provides an incumbent with a much smaller optimality gap at termination.

Table 1 attests to the solid performance of (**TR − A**) - **BDS** regardless of the range of the processing times. The performance indicators related to (**TR − A**) - **BDS** for both $p_{\max} = 20$ and $p_{\max} = 100$ are similar. We reckon that two factors are at play here. First, the magnitude of the processing times has no effect on the size of (**RMP**) and the number of job-to-machine assignments, and the pseudo-polynomial size of (**TR − A**) is therefore completely relegated to the dual slave problems. Second, the analytic solution of (**DS**$_k$) offsets the pseudo-polynomial size issue in practice.

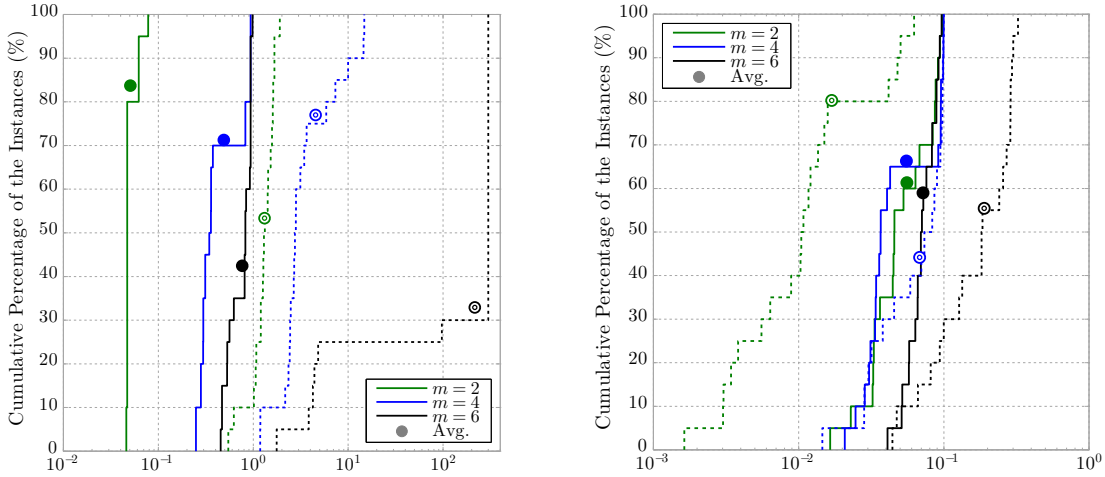
Next, we investigate how (**TR − A**) - **BDS** and (**CQ**) - **CPLEX** scale with the number of jobs and machines. For a fixed n, the solution times of (**TR − A**) - **BDS** and (**CQ**) - **CPLEX** increase with m. That is, both methods favor larger $\frac{n}{m}$ ratios. This may be regarded as a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

Figures 1-2 further substantiate the robustness and scalability of (**TR − A**) - **BDS** as an exact approach for *Rm-TWCT*. The empirical distributions of the solution times and the optimality gaps associated with both methods are depicted in these figures, where each curve is based on 20 instances. The horizontal axes are in logarithmic scale to increase the readability of the graph. The median solution times and optimality gaps are associated with the 50% mark on the vertical axis, and the average gaps are explicitly indicated. Note that the shape of the optimality gap curves to the left of the 10^{-1}% mark do not bear any meaning because the relative optimality gap parameter of `CPLEX` is set to `EpGap` $= 10^{-3} = 10^{-1}$%. The relative insensitivity of (**TR − A**) - **BDS** to n for a fixed m is also evident from Figure 1, where the curves for a fixed m are stacked on top of each other from Figure 1a toward Figure 1c. As stated previously, the number of machines is the main determinant of the solution time of (**TR − A**) - **BDS**; the curves for a fixed n shift from left to right as m increases. Furthermore, we can also claim that (**TR − A**) - **BDS** demonstrates a very consistent performance for these instances because the solution time curve for a given (n, m) combination rises sharply and exhibits little variability across instances. Figure 1 confirms that the solution time performance of (**TR − A**) - **BDS** is superior to that of (**CQ**) - **CPLEX** because the curves for (**TR − A**) - **BDS** generally lie to the left of the corresponding curves for (**CQ**) - **CPLEX**. Figure 2 is less informative with respect to the solution times because both methods often hit the time limit for these instances. However, the optimality gap curves of (**TR − A**) - **BDS** clearly dominate those of (**CQ**) - **CPLEX**. Overall, we may draw the conclusion that (**TR − A**) - **BDS** is a scalable exact algorithm for *Rm-TWCT* and does either find the optimal solution faster than the current state-of-the-art in the literature or it identifies better incumbents at termination.

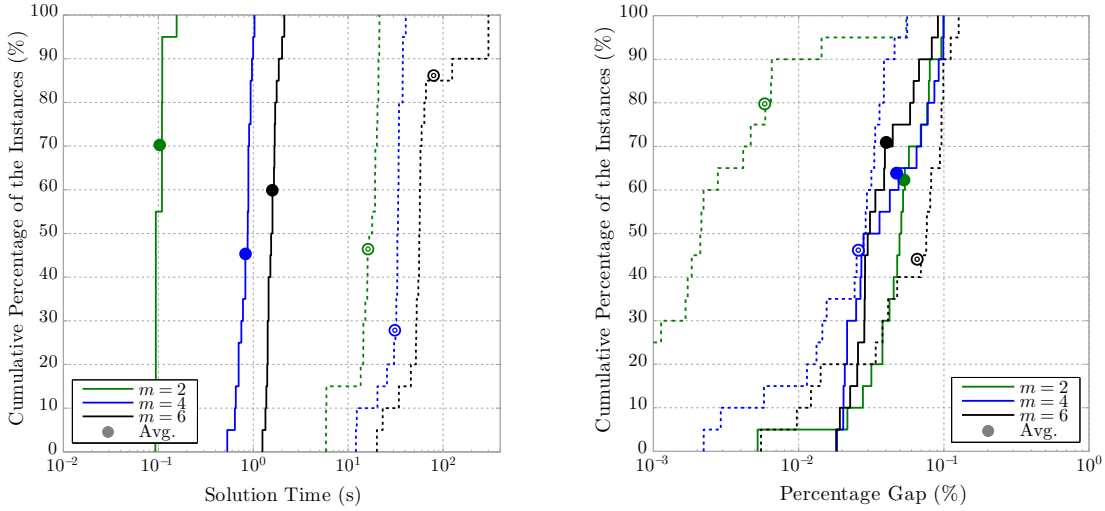
5. Conclusions and Future Research In this paper, we tackled the fundamental parallel machine scheduling problem *Rm-TWCT* which has been attacked by a variety of methodologies since the early 1970s. In a



(a) n = 100

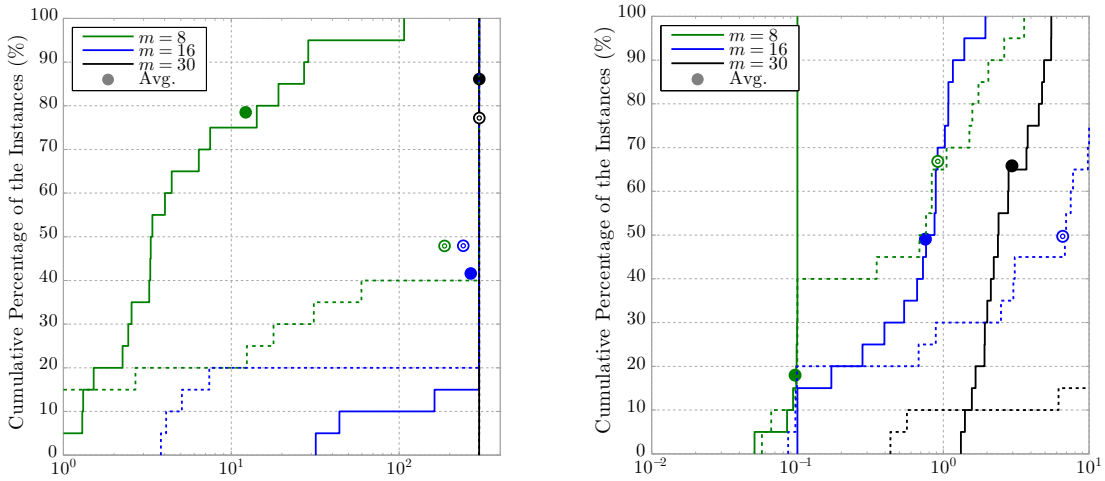


(b) n = 400

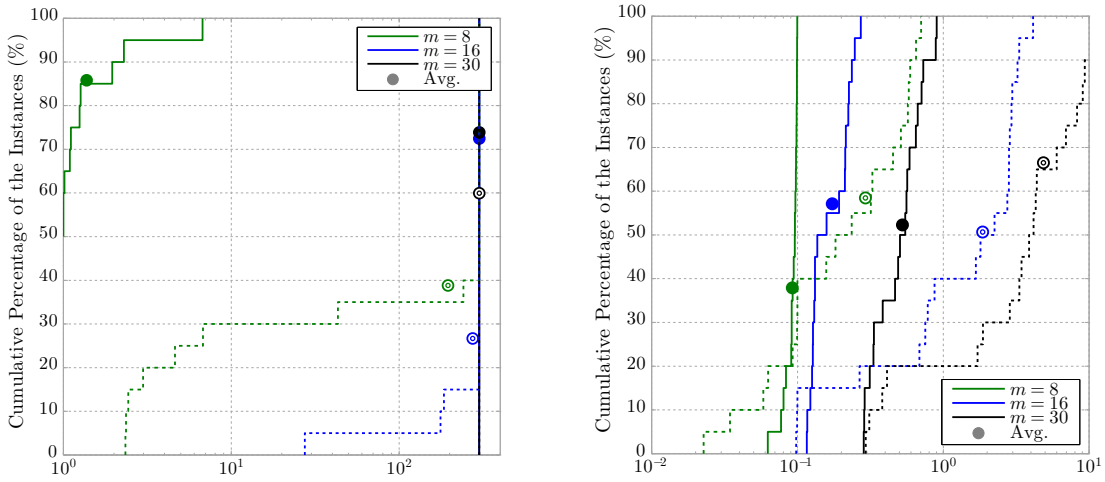


(c) n = 1000

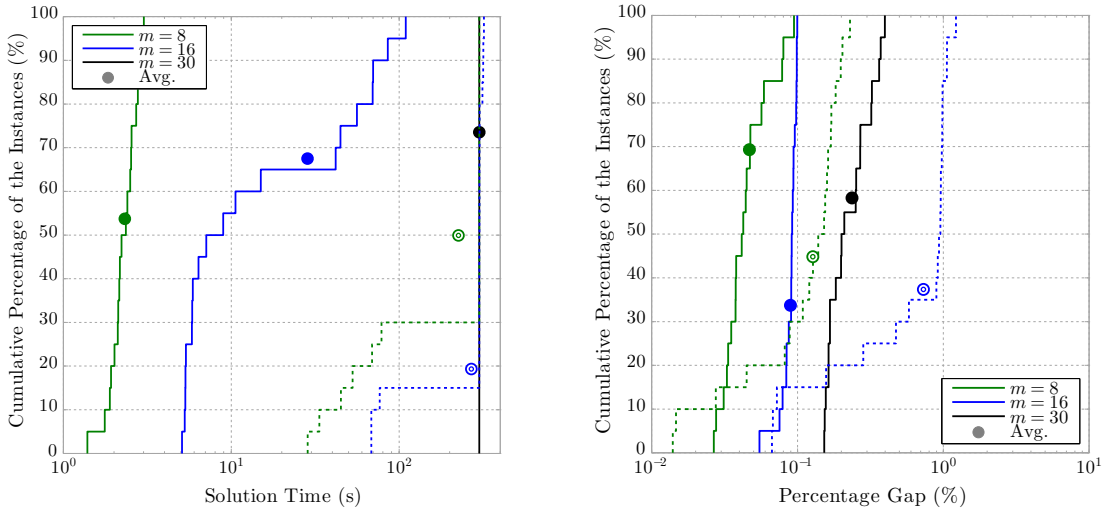
Figure 1 The empirical distributions of the solution times (on the left) and the optimality gaps (on the right) of (**TR − A**) - **BDS** (—) and (**CQ**) - **CPLEX** (– –) for *Rm-TWCT* instances with 2, 4, and 6 machines.



(a) $n = 100$



(b) $n = 400$



(c) $n = 1000$

Figure 2 The empirical distributions of the solution times (on the left) and the optimality gaps (on the right) of (**TR − A**) - **BDS** (—) and (**CQ**) - **CPLEX** (– –) for *Rm-TWCT* instances with 8, 16, and 30 machines.

field dominated by custom B&B methods, approximation algorithms, and (meta-)heuristics, our approach makes elegant use of generic mathematical programming techniques. We refrain from a traditional and compact modeling approach based on the job completion time variables and provide a new exact formulation of pseudo-polynomial size. Our formulation for *Rm-TWCT* is amenable to Benders decomposition, and we devise a computationally very effective algorithm that incorporates analytic solutions for the dual slave problems and a speedy cut strengthening procedure. The end product is a fast and scalable exact algorithm for *Rm-TWCT* which may even be employed as a subroutine in iterative decomposition-based algorithms developed for more complex shop scheduling problems.

Along with the more traditional time-indexed formulations, this study demonstrates that moving away from the natural space of variables to an extended variable space may help attain better formulations and algorithms for machine scheduling problems. A research question worth investigating in the future is to explore scheduling problems in other domains that may benefit from similar techniques.

References

Azizoglu, M. and Kirca, O. (1999a). On the minimization of total weighted flow time with identical and uniform parallel machines. *Eur J Oper Res*, 113(1):91–100. (Cited on pages 3 and 4.)

Azizoglu, M. and Kirca, O. (1999b). Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. *IIE Trans*, 31(2):153–159. (Cited on pages 3, 4, and 10.)

Barnes, J. W. and Brennan, J. (1977). An improved algorithm for scheduling jobs on identical machines. *AIIE Transactions*, 9(1):25–31. (Cited on page 3.)

Belouadah, H. and Potts, C. N. (1994). Scheduling identical parallel machines to minimize total weighted completion time. *Discrete Appl Math*, 48(3):201–218. (Cited on page 3.)

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numer Math*, 4(1):238–252. (Cited on pages 2 and 6.)

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

Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Weglarz, J. (2007). *Handbook on scheduling: from theory to applications*. Springer. (Cited on page 3.)

Bruno, J., Coffman Jr, E. G., and Sethi, R. (1974). Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17(7):382–387. (Cited on page 2.)

Bülbül, K., Kaminsky, P., and Yano, C. (2007). Preemption in single machine earliness/tardiness scheduling. *J Sched*, 10(4-5):271–292. (Cited on pages 5 and 7.)

Burkard, R., Dell'Amico, M., and Martello, S. (2009). *Assignment Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. (Cited on pages 7 and 8.)

Chekuri, C. and Khanna, S. (2004). Approximation algorithms for minimizing average weighted completion time. In Leung, J. Y., editor, *Handbook of scheduling: algorithms, models, and performance analysis*. CRC Press, Boca Raton, FL, USA, 2004. (Cited on page 3.)

Chen, Z.-L. and Powell, W. B. (1999). Solving parallel machine scheduling problems by column generation. *INFORMS J Comput*, 11(1):78–94. (Cited on pages 4, 5, 13, and 15.)

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

Detienne, B., Dauzère-Pérès, S., and Yugma, C. (2011). Scheduling jobs on parallel machines to minimize a regular step total cost function. *J Sched*, 14:523–538. (Cited on page 1.)

Dyer, M. and Wolsey, L. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Appl Math*, 26(2–3):255–270. (Cited on pages 4 and 8.)

Elmaghraby, S. E. and Park, S. H. (1974). Scheduling jobs on a number of identical machines. *AIIE transactions*, 6(1):1–13. (Cited on page 3.)

Fischetti, M., Salvagnin, D., and Zanette, A. (2010). A note on the selection of Benders' cuts. *Math Program*, 124(1-2):175–182. (Cited on page 11.)

Goemans, M. X., Queyranne, M., Schulz, A. S., Skutella, M., and Wang, Y. (2002). Single machine scheduling with release dates. *SIAM J Discrete Math*, 15(2):165–192. (Cited on page 8.)

Graham, R., Lawler, E., Lenstra, J., and Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E. J. and Korte, B., editors, *Discrete Optimization II*, volume 5 of *Ann Discrete Math*, pages 287–326. Elsevier. (Cited on page 2.)

IBM ILOG CPLEX (2012). IBM ILOG CPLEX Optimization Studio 12.5 Information Center. http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r5/index.jsp. Last viewed on 08/04/2014. (Cited on pages 10 and 13.)

Kedad-Sidhoum, S., Solis, Y. R., and Sourd, F. (2008). Lower bounds for the earliness–tardiness scheduling problem on parallel machines with distinct due dates. *Eur J Oper Res*, 189(3):1305–1316. (Cited on pages 5 and 6.)

Lawler, E. L. and Moore, J. M. (1969). A functional equation and its application to resource allocation and sequencing problems. *Manage Sci*, 16(1):77–84. (Cited on page 3.)

Lee, C.-Y. and Uzsoy, R. (1992). A new dynamic programming algorithm for the parallel machines total weighted completion time problem. *Oper Res Lett*, 11(2):73–75. (Cited on page 3.)

Li, K. and Yang, S.-l. (2009). Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied mathematical modelling*, 33(4):2145–2158. (Cited on pages 3 and 4.)

Lin, Y., Pfund, M., and Fowler, J. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Comput Oper Res*, 38(6):901–916. (Cited on page 3.)

Magnanti, T. L. and Wong, R. T. (1981). Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Oper Res*, 29(3):464–484. (Cited on page 11.)

Mokotoff, E. (2001). Parallel machine scheduling problems: a survey. *Asia Pacific Journal of Operational Research*, 18(2):193–242. (Cited on page 3.)

Nessah, R., Yalaoui, F., and Chu, C. (2008). A branch-and-bound algorithm to minimize total weighted completion time on identical parallel machines with job release dates. *Computers & Operations Research*, 35(4):1176–1190. (Cited on page 4.)

Pan, Y. and Shi, L. (2007). On the equivalence of the max-min transportation lower bound and the time-indexed lower bound for single-machine scheduling problems. *Math Program*, 110(3):543–559. (Cited on pages 5 and 6.)

Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer, 3rd edition. (Cited on pages 1 and 3.)

Plateau, M.-C. and Rios-Solis, Y. A. (2010). Optimal solutions for unrelated parallel machines scheduling problems using convex quadratic reformulations. *Eur J Oper Res*, 201(3):729–736. (Cited on pages 5 and 13.)

Posner, M. E. (1985). Minimizing weighted completion times with deadlines. *Operations Research*, 33(3):562–574. (Cited on page 8.)

Rodriguez, F., Blum, C., García-Martínez, C., and Lozano, M. (2012). GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times. *Ann Oper Res*, 201(1):383–401. (Cited on page 3.)

Rodriguez, F. J., Lozano, M., Blum, C., and García-Martínez, C. (2013). An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. *Comput Oper Res*, 40(7):1829–1841. (Cited on page 3.)

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

Sarin, S. C., Ahn, S., and Bishop, A. B. (1988). An improved branching scheme for the branch and bound procedure of scheduling n jobs on m parallel machines to minimize total weighted flowtime. *International Journal of Production Research*, 26(7):1183–1191. (Cited on page 3.)

Şen, H. and Bülbül, K. (2012). A simple, fast, and effective heuristic for the single-machine total weighted tardiness problem. In Demeulemeester, E. and Herroelen, W., editors, *Proceedings of the 13th Inter. Conf. on Project Management and Scheduling (PMS 2012)*, pages 282–286, Leuven, Belgium. (Cited on page 5.)

Şen, H. and Bülbül, K. (2015). A strong preemptive relaxation for weighted tardiness and earliness/tardiness problems

on unrelated parallel machines. *INFORMS J Comput*, 27(1):135–150. (Cited on pages 2, 3, 4, 5, 6, 10, 11, and 13.)

Shim, S.-O. and Kim, Y.-D. (2007). Minimizing total tardiness in an unrelated parallel-machine scheduling problem. *J Oper Res Soc*, 58(3):346–354. (Cited on page 1.)

Skutella, M. (2001). Convex quadratic and semidefinite programming relaxations in scheduling. *J ACM*, 48(2):206–242. (Cited on pages 4 and 5.)

Smith, W. E. (1956). Various optimizers for single-stage production. *Nav Res Log*, 3(1-2):59–66. (Cited on page 2.)

Sourd, F. and Kedad-Sidhoum, S. (2003). The one-machine problem with earliness and tardiness penalties. *J Sched*, 6(6):533–549. (Cited on page 5.)

Unlu, Y. and Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Comput Ind Eng*, 58(4):785–800. (Cited on page 4.)

van den Akker, J. M., Hoogeveen, J. A., and van de Velde, S. L. (1999). Parallel Machine Scheduling by Column Generation. *Oper Res*, 47(6):862–872. (Cited on pages 2, 4, 5, 13, and 15.)

Vredeveld, T. and Hurkens, C. (2002). Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. *INFORMS Journal on Computing*, 14(2):175–189. (Cited on pages 3 and 4.)

Yalaoui, F. and Chu, C. (2006). New exact method to solve the $Pm/r_j/\sum_j C_j$ schedule problem. *International Journal of Production Economics*, 100(1):168–179. (Cited on page 3.)

Appendix F. Proof of Proposition 3.1 Proof. The proof consists of two main steps. First, we show that $(\overline{\boldsymbol{u}}_k, \overline{\boldsymbol{v}}_k)$ is a feasible solution for $(\mathbf{DS_k - R})$, i.e., $\overline{v}_{kt} \leq 0$ for all $t = 1, \ldots, H_k$ and $\overline{u}_{jk} + \overline{v}_{kt} \leq c_{jkt}$ for all $j \in J_k$ and $t = 1, \ldots, H_k$. Then, we demonstrate that the objective function value associated with this feasible solution of the dual slave problem is equal to that of the optimal solution of the corresponding primal problem. Our focus in this proof is entirely on $(\mathbf{DS_k - R})$ for a given machine k, and therefore, every specific job or set of jobs referred to in the following belongs to J_k.

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

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

where

$$0 \leq r(t) = \sum_{i \leq l(t)} p_{ik} - t \leq p_{l(t)k} - 1, \tag{30}$$

and the weights and the processing times are strictly positive.

To show that $\overline{u}_{jk} + \overline{v}_{kt} \leq c_{jkt}$ is satisfied for all $j \in J_k$ and $t = 1, \ldots, H_k$, we substitute the values of $\overline{u}_{jk}$ and $\overline{v}_{kt}$ from (27) and c_{jkt} from (10). The constraint $\overline{u}_{jk} + \overline{v}_{kt} \leq c_{jkt}$ then reduces to

$$\begin{aligned} & \frac{w_j}{p_{jk}}\left(\sum_{i \leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2}\right) + \sum_{i > j} w_i + \frac{w_{l(t)}}{p_{l(t)k}}\left(t - \sum_{i \leq l(t)} p_{ik}\right) - \sum_{i > l(t)} w_i \quad && \leq \frac{w_j}{p_{jk}}\left(t + \frac{p_{jk}}{2} - \frac{1}{2}\right) \\ \iff & \frac{w_j}{p_{jk}}\left(\sum_{i \leq j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2}\right) + \sum_{i > j} w_i - \frac{w_{l(t)}}{p_{l(t)k}} r(t) - \sum_{i > l(t)} w_i && \leq \frac{w_j}{p_{jk}}\left(\sum_{i \leq l(t)} p_{ik} - r(t) + \frac{p_{jk}}{2} - \frac{1}{2}\right) \end{aligned} \tag{31}$$

by replacing $t - \sum_{i \leq l(t)} p_{ik}$ by $-r(t)$ and t by $\sum_{i \leq l(t)} p_{ik} - r(t)$ based on (30). In order to establish the validity of (31), we consider the cases $j \leq l(t)$ and $j > l(t)$ separately.

If $j \leq l(t)$, (31) simplifies to

$$\frac{w_j}{p_{jk}}\left(-\sum_{j < i \leq l(t)} p_{ik}\right) + \sum_{j < i \leq l(t)} w_i \leq r(t)\left(\frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}}\right). \tag{32}$$

Note that $j \leq l(t)$ implies $\frac{w_j}{p_{jk}} \geq \frac{w_{l(t)}}{p_{l(t)k}}$ and leads to $\left(p_{l(t)k} - 1\right)\left(\frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}}\right) \leq r(t)\left(\frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}}\right)$ based on (30). Therefore,

(32) holds if

$$-\sum_{j<i\le l(t)} p_{ik}\frac{w_j}{p_{jk}} + \sum_{j<i\le l(t)} p_{ik}\frac{w_i}{p_{ik}} \le \left(p_{l(t)k}-1\right)\left(\frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}}\right) \tag{33}$$

$$\iff \sum_{j<i<l(t)} p_{ik}\left(\frac{w_i}{p_{ik}} - \frac{w_j}{p_{jk}}\right) \le \frac{w_j}{p_{jk}} - \frac{w_{l(t)}}{p_{l(t)k}} \tag{34}$$

is satisfied, where the transition from (33) to (34) requires adding $p_{l(t)k}\frac{w_j}{p_{jk}} - w_{l(t)}$ to both sides of (33). The inequality (34) is clearly correct since $\left(\frac{w_i}{p_{ik}} - \frac{w_j}{p_{jk}}\right) \le 0$ for $i \ge j$ and $\left(\frac{w_j}{p_{jk}} - \frac{w_{l(t)}}{p_{l(t)k}}\right) \ge 0$, and this completes the argument for the first case with $j \le l(t)$.

If $j > l(t)$, re-arranging the terms of (31) leads to

$$\frac{w_j}{p_{jk}}\left(\sum_{l(t)<i\le j} p_{ik}\right) - \sum_{l(t)<i\le j} w_i \le r(t)\left(\frac{w_{l(t)}}{p_{l(t)k}} - \frac{w_j}{p_{jk}}\right). \tag{35}$$

The inequality $\frac{w_j}{p_{jk}} \le \frac{w_{l(t)}}{p_{l(t)k}}$ follows from $j > l(t)$, and we conclude that the right hand side of (35) is non-negative. Therefore, in order to prove that (35) is satisfied it is sufficient to demonstrate the correctness of this relation:

$$\sum_{l(t)<i\le j} p_{ik}\frac{w_j}{p_{jk}} - \sum_{l(t)<i\le j} p_{ik}\frac{w_i}{p_{ik}} = \sum_{l(t)<i\le j} p_{ik}\left(\frac{w_j}{p_{jk}} - \frac{w_i}{p_{ik}}\right) \le 0. \tag{36}$$

The validity of inequality (36) derives from $\left(\frac{w_j}{p_{jk}} - \frac{w_i}{p_{ik}}\right) \le 0$ for $i \le j$. This yields the correctness of (31) for the second case with $j > l(t)$, and $(\overline{u}_k, \overline{v}_k)$ is certified as a feasible solution of $(\mathbf{DS_k - R})$.

As noted before, the optimal schedule of the restricted cut generation subproblem $(\mathbf{TR_k - R})$ – the primal problem associated with $(\mathbf{DS_k - R})$ – follows the WSPT order for the jobs in J_k. Therefore, the associated optimal objective function value is calculated as $\sum_{j\in J_k} w_j\left(\sum_{i\le j} p_{ik}\right)$, where the completion time of job j in the non-preemptive WSPT schedule is equal to the sum of the processing times of the jobs placed earlier in the WSPT sequence. Thus, in order to complete the proof, we must argue that the objective function value associated with $(\overline{u}_k, \overline{v}_k)$ in $(\mathbf{DS_k - R})$ is equal to $\sum_{j\in J_k} w_j\left(\sum_{i\le j} p_{ik}\right)$.

$$\sum_{j\in J_k} p_{jk}\overline{u}_{jk} + \sum_{t=1}^{H_k}\overline{v}_{kt} = \sum_{j\in J_k} p_{jk}\left(\frac{w_j}{p_{jk}}\left(\sum_{i\le j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2}\right) + \sum_{i>j} w_i\right) + \sum_{t=1}^{H_k}\left(\frac{w_{l(t)}}{p_{l(t)k}}\left(t - \sum_{i\le l(t)} p_{ik}\right) - \sum_{i>l(t)} w_i\right) \tag{37}$$

$$= \sum_{j\in J_k} p_{jk}\left(\frac{w_j}{p_{jk}}\left(\sum_{i\le j} p_{ik} + \frac{p_{jk}}{2} - \frac{1}{2}\right) + \sum_{i>j} w_i\right) + \sum_{j\in J_k}\left(\frac{w_j}{p_{jk}}\left(-\sum_{i=0}^{p_{jk}-1} i\right) - p_{jk}\sum_{i>j} w_i\right) \tag{38}$$

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

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

For the transition from (37) to (38), observe that each job $j \in J_k$ becomes the job $l(t)$ for p_{jk} consecutive time periods, and the difference $\left(t - \sum_{i\le l(t)} p_{ik}\right)$ runs from $-(p_{jk}-1)$ to zero during these time periods. Therefore, we have $\sum_{t=1}^{H_k}\frac{w_{l(t)}}{p_{l(t)k}}\left(t - \sum_{i\le l(t)} p_{ik}\right) = \sum_{j\in J_k}\frac{w_j}{p_{jk}}\left(-\sum_{i=0}^{p_{jk}-1} i\right)$. A similar argument yields $-\sum_{t=1}^{H_k}\sum_{i>l(t)} w_i = -\sum_{j\in J_k} p_{jk}\sum_{i>j} w_i$. □

Appendix G. Benders Decomposition Algorithm with Cut Strengthening for (TR − A)

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

Create (**RMP**) with (22), (23), (25), and the load balancing constraints (26); `// Initialization.`
Invoke `CPLEX` on (**RMP**); `// Main loop.`
repeat
  Identify a new candidate incumbent solution $\overline{y}$ with an objective value of $\sum_{k=1}^{m} \overline{\eta}_k$;
  accept_candidate = true;
  $[cuts, z_1(\overline{y}), \ldots, z_m(\overline{y})] = generate\_cuts(\overline{y})$; `// cuts is a collection of m cuts.`
  for $k = 1$ *to* m **do**
    if $\overline{\eta}_k < z_k(\overline{y})$ **then** `// ` $\overline{y}$ `violates a missing Benders cut.`
      Add $cuts_k$ to (**RMP**) as a lazy constraint, *accept_candidate* = false;
  end
until *CPLEX determines that the relative optimality gap of the current incumbent is less than some threshold*;
The best available job partition $\overline{y}^*$ for (**TR − A**) is retrieved from `CPLEX`. The optimal solution for *Rm-TWCT* is obtained by applying the WSPT rule independently to the set of jobs on each machine;

Algorithm 2: Procedure *generate_cuts*.

input : A feasible partition $\overline{y}$ of the jobs to the machines.
output: Returns $z_k(\overline{y})$ and a strengthened Benders cut for all machines $k = 1, \ldots, m$.

for $k = 1$ *to* m **do**
  Compute the optimal solution $(\overline{u}_k, \overline{v}_k)$ of (**DS$_k$ − R**) as given in (27) and calculate $z_k(\overline{y})$;
  Retrieve the job completion times $C_j,\ j \in J_k$, in the associated optimal solution of (**TR$_k$ − R**);
  $[(\overline{u}'_k, \overline{v}'_k)] = strengthen\_cut(J_k, (C_j,\ j \in J_k), (\overline{u}_k, \overline{v}_k))$ is an optimal solution of (**DS$_k$**);
  Generate a strengthened Benders cut of the form (24) from $(\overline{u}'_k, \overline{v}'_k)$ and add to *cuts*;
end

Algorithm 3: Procedure *strengthen_cut*.

input : J_k – Jobs assigned to machine k, re-labeled in WSPT order,
  $C_j,\ j \in J_k$ – Job completion times in the optimal solution of (**TR$_k$ − R**),
  $(\overline{u}_k, \overline{v}_k)$ – Optimal solution of (**DS$_k$ − R**) as specified in (27).
output: $(\overline{u}'_k, \overline{v}'_k)$.

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