Computers & Industrial Engineering 58 (2010) 785–800

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems ☆

Yasin Unlu, Scott J. Mason *

Department of Industrial Engineering, University of Arkansas, 4207 Bell Engineering Center, Fayetteville, AR 72701, USA

ARTICLE INFO

Article history:
Received 4 August 2009
Received in revised form 10 February 2010
Accepted 13 February 2010
Available online 18 February 2010

Keywords:
Mixed integer programming
Non-preemptive
Parallel machine scheduling

ABSTRACT

Mixed integer programming (MIP) formulations for scheduling problems can be classified based on the decision variables upon which they rely. In this paper, four different MIP formulations based on four different types of decision variables are presented for various parallel machine scheduling problems. The goal of this research is to identify promising optimization formulation paradigms that can subsequently be used to either (1) solve larger practical scheduling problems of interest to optimality and/or (2) be used to establish tighter lower solution bounds for those under study. We present the computational results and discuss formulation efficacy for total weighted completion time and maximum completion time problems for the identical parallel machine case.

1. Introduction

The field of deterministic scheduling has received a great deal of research attention in the last 40 years. Scheduling research has been motivated by questions that arise in production planning, telecommunications, logistics, and computer control, to name only a few application domains. Manufacturing and service industries use many different solution approaches to schedule and/or sequence activities. As an organization's failure to meet customer due date/delivery requirements can result in a significant loss of goodwill, practitioners and researchers have investigated numerous types of scheduling problems in the hopes of producing optimal or near-optimal solution methodologies. Indeed, the field of machine scheduling has been and will continue to be a rich and promising field for research in the future.

A machine can be thought of as a production resource on which activities (jobs) are sequenced and processed. Often, various sequencing and/or processing restrictions exist against which decision makers try to minimize some performance measure(s) (objective function(s)) of interest. While many different types and classes of scheduling problems exist, non-preemptive scheduling of multiple machines operating in parallel ("parallel machines") is the focus of this paper. The non-preemptive, parallel machine scheduling problem is to sequence jobs without interrupting (preempting) job processing at any time across all available machines. These machines may or may not be identical to one another. Parallel machine job processing environments can be described as identical machines operating in parallel (identical machine environment: "*Pm*"), machines operating in parallel at different (albeit consistent) speeds/processing rates (non-identical machine environment: "*Qm*"), and machines operating in parallel with job-dependent processing speeds (unrelated machine environment: "*Rm*").

These three parallel machine processing environments can be investigated over a wide range of objective functions, each combination of which results in a different scheduling problem. For example, an important objective function in the scheduling literature is to minimize total weighted completion time for all jobs: $\sum_j w_j C_j$, where weight w_j describes job j's priority or importance and C_j represents job j's completion time. Job due date-based performance measures, such as the scheduling problem with the objective function of minimax type (e.g. maximum lateness: $\min L_{\max} = \min[\max_j(C_j - d_j)]$) and minsum type (e.g. total weighted tardiness: $\min \sum_j w_j T_j = \min \sum_j w_j \max(C_j - d_j, 0)$) are also of research and practical importance.

The scheduling literature contains a wide array of heuristic solution procedures for analyzing production scheduling problems, as the majority of practically motivated production scheduling problems are NP hard. However, often a researcher's first key step prior to developing an effective heuristic is to develop a mathematical programming formulation of the scheduling problem under study. This is often done to understand the underlying structure of the problem (if any exists) and as a means to evaluate the efficacy of subsequently developed heuristics by comparing heuristic solutions to know optimal solutions for small, solvable problem sets.

Although mathematical programming models have been developed by a number of researchers, they typically do not perform very well for practically motivated problem instances due to model

☆ This manuscript was processed by Area Editor Maged M. Dessouky.

* Corresponding author. Tel.: +1 479 575 5521; fax: +1 479 575 8431.
E-mail address: mason@uark.edu (S.J. Mason).

0360-8352/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2010.02.012

formulation and/or computational difficulties. However, recent computational performance advances have become available due to increasingly powerful microprocessors becoming available to the public. In addition, researchers have begun to exploit the polyhedral properties of some optimization models, including production scheduling models, in order to develop more efficient, effective combinatorial optimization solution techniques.

Optimization formulations and relaxed formulations are proven methods for producing strong lower bounds for many types of scheduling problems. These bounds may eventually lead to the development of an effective solution strategy or some other type of algorithmic approach for solving production scheduling problems. However, many different types of optimization model formulation paradigms exist, each of which may potentially have its own strengths and weaknesses in terms of the types of scheduling problems that lend themselves to analysis using a given paradigm.

Although a number of machine scheduling survey papers have appeared in literature, only a few of them focus on optimization model formulation paradigms. First, Blazewicz, Dror, and Weglarz (1991) compile a large number of mathematical formulations for and provide the complexity of various production scheduling problems. Another comprehensive survey of mixed integer programming (MIP) formulations is given by Queyranne and Schulz (1994), who also describe the polyhedral analysis of some single machine scheduling problem formulations. Finally, Keha, Khowala, and Fowler (2009) present a comparative analysis of the computational performance for the single machine formulations in Queyranne and Schulz (1994).

As no existing research is evident that investigates parallel machine optimization model formulation paradigms, we compare and contrast mathematical programming models for parallel machine scheduling problems for important performance measures. Clearly, successful investigation into various parallel machine environments can be easily reduced to the single machine environment. The research goal is to identify promising optimization formulation paradigms for various types of parallel machine and objective function environments that can subsequently be used to either (1) solve larger practical scheduling problems of interest to optimality and/or (2) be used to establish tighter lower solution bounds for large scheduling problems under study.

2. Production scheduling optimization modeling paradigms

Scheduling problem MIP formulations can be classified or described based on the decision variables upon which they rely. A number of different types of decision variables exist with which researchers previously have modeled and optimized production scheduling problems—these are described in detail in this section.

2.1. Job completion time variables

Job completion time is a key metric in assessing the quality of a proposed production schedule. In fact, a vast majority of scheduling problem performance measures is derived from and/or is a function of job completion times. Completion time variables, which are sometimes referred to as "natural date variables" (Queyranne & Schulz, 1994), were used by Balas (1985) in his disjunctive formulation of a well-known job shop scheduling problem. Later, the initial formulation of Balas (1985) was also studied by Queyranne and Wang (1991) and Queyranne (1993).

2.2. Assignment and positional date variables

MIP formulations containing assignment and positional date variables specify which job is scheduled next and at what time this job will start processing. For example, in a single machine scheduling problem, n jobs are assigned to n available positions. Lasserre and Queyranne (1992) consider a single machine problem as a system which can be controlled at discrete instants in time using a combination of discrete and continuous controls. Further, the problem of minimizing the total number of tardy jobs on a single machine was studied by Dauzère-Pérès (1997) and Dauzère-Pérès and Sevaux (2003) using assignment and positional date variables.

2.3. Linear ordering variables

The linear ordering of jobs can be thought of as the set of all sequence permutations (solutions). In this manner, a binary linear ordering variable is defined—it is typically equal to one if a given job succeeds another job, thereby describing precedence relationships among all jobs. Linear ordering variables are also referred to as "sequencing variables" (Pinedo, 2002). Dyer and Wolsey (1990) use linear ordering in combination with starting time variables in their single machine problem study with release dates. In addition to Dyer and Wolsey (1990) developing several polyhedral approaches, these variables also are studied by Blazewicz et al. (1991), Nemhauser and Savelsbergh (1992), and Chudak and Hochbaum (1999).

2.4. Time indexed variables

Time indexed variables, which typically assign jobs to time periods, are decision variables indexed by both job and time dimensions that have been used in a variety of machine scheduling problems. Time indexed variables for machine scheduling were first introduced by Sousa and Wolsey (1992) who study a single machine case. Later, different problems in a variety of single machine environments are studied by Blazewicz et al. (1991), Sousa and Wolsey (1992), Chan, Kaminsky, Muriel, and Simchi-Levi (1995), Van den Akker, Hurkens, and Savelsbergh (2000), Van den Akker, Van Hoesel, and Savelsbergh (1999), and Šoric (2000).

2.5. Network variables

Network or "traveling salesman variables" (Queyranne & Schulz, 1994) initially were used to model the single machine scheduling problem with sequence dependent processing times, as it was shown to resemble a time-dependent traveling salesman problem (TSP) (Houck & Vemuganti, 1976). Precedence relationships for the same problem studied by Queyranne and Schulz (1994) afford polyhedral analysis of the model. Blazewicz et al. (1991) give a network formulation based on Miller, Tucker, and Zemlin (1960) for single machine scheduling problems, also extending the formulation to consider job release dates/times. Cakici and Mason (2007) present a network-based MIP formulation for parallel machine scheduling in the presence of auxiliary resource constraints.

3. Optimization models

This section contains discrete optimization formulations for parallel machine scheduling problems using four different modeling (decision variable) paradigms. Specifically, the decision variables considered are time indexed variables, network variables, and assignment and positional date variables. Job completion time variables are too simplistic to consider for the parallel machine environment case, especially considering their known poor performance in single machine environments. For this reason, this approach is not investigated in this paper.

The overall research objective of this paper is to determine if one or more modeling paradigms lend themselves to improved tractability for different parallel machine scheduling problems of

interest. As the problems under study are all NP hard, improved model tractability can help researchers evaluate the performance of proposed heuristics for larger problem instances and/or provide tighter lower bounds for parallel machine scheduling problems of interest. In order to increase our computational efficacy, we also extend polyhedral cut schemes from the single and identical parallel machine scheduling literature to both the non-identical (*Qm*) and the unrelated parallel machine (*Rm*) environments in this paper.

3.1. Assumptions and notation

In all parallel machine scheduling problem cases, we assume n jobs are to be processed without preemption. We further assume that all parameters are known and given in integer values. The following notation describes the sets, parameters, and variables used in the mathematical formulations that follow (note: not all notation is required in each model):

Sets

J	set of jobs, indexed $j = 1,\ldots,n$
μ	set of machines, indexed $i = 1,\ldots,m$
τ	set of time periods, indexed $t = 1,\ldots,l$

Parameters

d_j	due date of job j
r_j	release date of job j
w_j	priority or weight of job j
v_i	speed of machine i; in the case of job-dependent machine speed, v_{ij} denotes the speed of machine i for job j
p_j^i	processing time of job j on machine i, defined as p_j for the *Pm* environment, $\frac{p_j}{v_i}$ for the *Qm* case, and $\frac{p_j}{v_{ij}}$ for the *Rm* environment
l	upper bound for makespan (total time required to process all jobs), defined for different machine environments as follows

Identical parallel machines (Pm): $l \geqslant \sum_j p_j + \max_j\{r_j\}$

Non-identical parallel machines (Qm):

$$l \geqslant \max_i \left\{ \frac{\sum_j p_j}{v_i} + \max_j\{r_j\} \right\}$$

Unrelated parallel machines (Rm): $l \geqslant \max_i \left\{ \sum_j \frac{p_j}{v_{ij}} + \max_j\{r_j\} \right\}$

These three inequalities tighten the formulations which rely on knowledge of the completion time of the last job on a machine (i.e., upper bound of makespan). Similarly, we develop three different methods for bounding our penalty parameter:

M a large number (i.e., "big-M")

Pm: $M \geqslant \sum_j p_j + \max_j\{r_j\}$

Qm: M_i can be defined for each machine i as $M_i \geqslant \frac{\sum_j p_j}{v_i} + \max_j\{r_j\}$. However, in order to promote tighter model formulations, we define parameter $M_{ij} = M_i - \left(\frac{p_j}{v_i} + r_j\right)$ for each job j and machine i.

Rm: M_i can be defined for each machine i as $M_i \geqslant \sum_j \frac{p_j}{v_{ij}} + \max_j\{r_j\}$. Further, parameter M_{ij} for each job j and machine i can be defined as follows: $M_{ij} = M_i - \left(\frac{p_j}{v_{ij}} + r_j\right)$

Decision variables

C_j	Completion time of job j; non-negative; used in minimizing total weighted completion time: $\sum_{j\in J} w_j C_j$
L_j	Lateness of job $L_j = C_j - d_j$; used in minimizing maximum lateness: $L_{\max} = \max_j(C_j - d_j)$
T_j	Tardiness of job $T_j = \max\{0, C_j - d_j\}$; used in minimizing total weighted tardiness: $\sum_{j\in J} w_j T_j$
$U_j = 1$	if job $j$is tardy; otherwise, = 0; used in minimizing total number of tardy jobs: $\sum_{j\in J} U_j$

3.2. Time indexed model for scheduling parallel machines (M1)

In a time indexed model formulation, a time horizon is discretized into time periods $1,\ldots,l$, where l is an upper bound of the last job's completion time (i.e., makespan). Let binary variable $\chi_{ij}^t = 1$ if job $j \in J$ starts processing on machine $i \in \mu$ at time $t \in \tau$; otherwise, $\chi_{ij}^t = 0$.

3.2.1. Model M1 constraints

Constraint set (1) ensures that each job starts processing on only one machine at only one point in time, while constraint set (2) allows at most one job to be processed at any time on any machine.

$$\sum_{i\in\mu} \sum_{t=0}^{l-1} \chi_{ij}^t = 1 \quad j \in J \tag{1}$$

$$\sum_{j\in J} \sum_{h=\max\left(0,t-p_j^i\right)}^{t-1} \chi_{ij}^h \leqslant 1 \quad i \in \mu,\ t = 1,\ldots,l \tag{2}$$

Job j's completion time C_j is given by constraint set (3):

$$C_j \geqslant \sum_{i\in\mu} \sum_{t=0}^{l-1} \left(t + p_j^i\right) \chi_{ij}^t \quad j \in J \tag{3}$$

We previously define job processing time p_j^i so as to take care of all three parallel machine environments. In model M1, p_j^i denotes the processing time of job j on machine i. If identical parallel machines are present (*Pm*), then all p_j^i parameters are equal for each machine and every job. In the *Qm* environment, all p_j^i parameters are determined by machine speed v_i. Finally, p_j^i values are a function of v_{ij} (i.e., both machine speed and job) in the *Rm* environment.

3.2.2. Model M1 for different objective functions

Total weighted completion time: $\min \sum_{j\in J} w_j C_j$, subject to (1)–(3)

Total weighted tardiness: $\min \sum_{j\in J} w_j T_j$, subject to (1)–(4):

$$T_j \geqslant C_j - d_j \quad j \in J \tag{4}$$

Maximum lateness: $\min L_{\max}$, subject to (1)–(3) and (5):

$$L_{\max} \geqslant C_j - d_j \quad j \in J \tag{5}$$

Total number of tardy jobs: $\min \sum_{i\in\mu} \sum_{j\in J} \sum_{t=0}^{l-1} \frac{\max[0, t-d_j+p_j]}{\max[1, t-d_j+p_j]} \chi_{ij}^t$ subject to (1) and (2).

3.2.3. Incorporating ready times in model M1

It should be noted that job release dates can be incorporated into model M1, regardless of the objective function of interest, by adding the following constraint set:

$$\sum_{i\in\mu}\sum_{t=0}^{r_j-1}\chi_{ij}^{t}=0 \quad j\in J \tag{6}$$

3.3. Network model for scheduling parallel machines (M2)

Network-based model formulations have proven quite useful for modeling a great variety of problems. While the network model for the single machine scheduling problem can be regarded as a TSP, parallel machine scheduling problems relate to the capacitated vehicle routing problem (CVRP) when (1) the jobs to be scheduled are modeled as customers (nodes) and (2) the machines represent the vehicles being routed. In this regard, the capacity of each vehicle can be a surrogate measure for an upper bound on the completion time of the last job processed on each machine. Each "route" defines a machine's schedule, with a dummy node being added to designate the starting and ending node of each vehicle's route (machine's sequence). As the dummy job is scheduled twice on the machine (i.e., at the head and tail of the job sequence), its processing time is set to 0. Further, employing network formulations for machine scheduling problems allows for taking advantage of recent research advances by adding polyhedral cuts to the model originally derived for the TSP to improve model tractability.

In order to develop a network model for the general parallel machine scheduling problem, we extend the formulation of Cakici and Mason (2007) to the *Qm* and *Rm* environments. Let binary variable $x_{ij}^{i}=1$ if job $\imath$ is processed immediately before job j on machine i; otherwise, $x_{ij}^{i}=0$.

3.3.1. Model M2 constraints

Constraint sets (8) and (9) ensure that at most one job is scheduled just before and after dummy job 0, respectively.

$$\sum_{i\in\mu}x_{i0}^{i}\leqslant 1 \quad \imath\in J:\imath\neq 0 \tag{7}$$

$$\sum_{i\in\mu}x_{0j}^{i}\leqslant 1 \quad j\in J:j\neq 0 \tag{8}$$

Constraint sets (10) and (11) guarantee that all jobs are scheduled on a machine.

$$\sum_{\imath\in J}\sum_{i\in\mu}x_{ij}^{i}=1 \quad j\in J:j\neq 0,\ \ \imath\neq j \tag{9}$$

$$\sum_{j\in J}\sum_{i\in\mu}x_{ij}^{i}=1 \quad \imath\in J:j\neq 0,\ \ \imath\neq j \tag{10}$$

Constraint set (12) ensures that each job, except the dummy job, has exactly one predecessor and one successor, thereby imposing flow continuity and connectivity at every node for each tour.

$$\sum_{\imath\in J:\imath\neq j}x_{ij}^{i}-\sum_{\imath\in J:\imath\neq j}x_{j\imath}^{i}=0 \quad j\in J:j\neq 0,\ \ i\in\mu \tag{11}$$

Constraint sets (8)–(12) are the degree constraints presented in most node-based network formulations prior to the specification of sub-tour elimination constraints.

The bookkeeping constraint sets that calculate job completion times and eliminate any possibility of sub-tours can be constructed based on sub-tour elimination constraints developed for the CVRP. The following sub-tour elimination constraint set is based on constraints proposed by Kulkarni and Bhave (1985) (later corrected by Kara, Laporte, & Bektas (2004)) which are extensions of the original TSP sub-tour elimination constraints formulated by Miller et al. (1960):

$$C_{\imath}-C_j+(M-r_j)x_{ij}^{i}\leqslant M-\left(r_j+p_j^{i}\right)\ \ \forall\imath\in J,\forall j\in J:j\neq 0,\ \imath\neq j,\ i\in\mu \tag{12}$$

Note that this sub-tour elimination constraint set determines job completion times on each machine.

3.3.2. Model M2 for different objective functions

Total weighted completion time: $\min\sum_{j\in J}w_jC_j$, subject to (7)–(12)
Total weighted tardiness: $\min\sum_{j\in J}w_jT_j$, subject to (4), and (7)–(12)
Maximum lateness: $\min L_{\max}$, subject to (5), and (7)–(12)
Total number of tardy jobs: $\min\sum_{j\in J}U_j$, subject to (7)–(13):

$$C_j\leqslant d_j+MU_j \quad j\in J \tag{13}$$

3.3.3. Valid inequalities for model M2

In order to improve model tractability, the following constraint sets can be added to the network formulation model M2 for all three parallel machine environments:

$$C_{\imath}\geqslant r_{\imath}+\sum_{j\in J}\left(p_{\imath}^{i}+\max\left(0,r_j+p_j^{i}-r_{\imath}\right)\right)x_{j\imath}^{i}\ \ j\in J:j\neq 0,\ \imath\neq j,\ i\in\mu \tag{14}$$

$$C_{\imath}\leqslant M-\sum_{j\in J}\sum_{i\in\mu}p_j^{i}x_{ij}^{i}\ \ \imath\in J:\imath\neq 0,\ \imath\neq j \tag{15}$$

3.4. Assignment and positional date model for scheduling parallel machines (M3)

In an assignment and positional date model, decision variables are defined based on the notion that each machine has a fixed number of positions or slots into which jobs can be assigned. These positions by construction specify a job's relative position to all other jobs processed on the same machine and therefore, the job sequence on the machine. Let binary variables $u_{j\ell}^{i}=1$ if job j is assigned to position ℓ on machine i; otherwise, $u_{j\ell}^{i}=0$. Additionally, let non-negative positional date variable γ_{ℓ}^{i} denote the completion time of the job at position ℓ on machine i.

3.4.1. Model M3 constraints

Constraint set (16) ensures that all jobs are assigned to exactly one position on only one machine.

$$\sum_{i\in\mu}\sum_{\ell\in J}u_{j\ell}^{i}=1 \quad j\in J \tag{16}$$

Constraint set (17) guarantees that each position on every machine contains at most one job.

$$\sum_{j\in J}u_{j\ell}^{i}\leqslant 1 \quad \ell\in J,\ \ i\in\mu \tag{17}$$

The completion time of the job at position ℓ on each machine is determined by constraint sets (18) and (19).

$$\gamma_1^{i}\geqslant\sum_{j\in J}p_ju_{j1}^{i} \quad i\in\mu \tag{18}$$

$$\gamma_{\ell}^{i}\geqslant\gamma_{\ell-1}^{i}+\sum_{j\in J}p_j^{i}u_{j\ell}^{i} \quad i\in\mu,\ \ \ell\in J:\ell\geqslant 2 \tag{19}$$

Finally, the completion time of job j (regardless of its position) is given by constraint set (20).

$$C_j\geqslant\gamma_{\ell}^{i}-M\left(1-u_{j\ell}^{i}\right) \quad j\in J,\ \ \ell\in J,\ \ i\in\mu \tag{20}$$

The above constraint sets are extended from the original single machine formulation of Lasserre and Queyranne (1992). Queyranne and Schulz (1994) later studied the polyhedral structure of the single machine formulation, while Keha et al. (2009) propose two additional valid inequalities for the single machine case.

3.4.2. Model M3 for different objective functions

Total weighted completion time: $\min \sum_{j\in J} w_j C_j$, subject to (16)–(20)
Total weighted tardiness: $\min \sum_{j\in J} w_j T_j$, subject to (4), and (16)–(20)
Maximum lateness: $\min L_{\max}$, subject to (5), and (16)–(20)
Total number of tardy jobs: $\min \sum_{j\in J} U_j$, subject to (13), and (16)–(20)

3.4.3. Incorporating ready times in model M3

Job release dates are incorporated into model M3 by adding the following constraint set:

$$\gamma^i_\ell \geqslant \sum_{j\in J} \left(p^i_j + r_j\right) u^i_{j\ell} \quad i \in \mu,\ \ell \in J \tag{21}$$

3.5. Linear ordering model for scheduling parallel machines (M4)

The linear ordering model is based on binary linear ordering variable $\delta_{\iota j}$ which equals 1 whenever job ι precedes job j; otherwise, $\delta_{\iota j} = 0$. Note that job ι is not necessarily positioned immediately before job j when $\delta_{\iota j} = 1$. Additionally, let binary variable $z_{\iota i} = 1$ if job ι is positioned on machine ι (otherwise, $z_{\iota i} = 0$) and binary variable $y_{\iota j} = 1$ if jobs ι and j are *not* scheduled on the same machine (otherwise, $y_{\iota j} = 0$).

3.5.1. Model M4 constraints

Constraint set (22) ensures that either job ι is positioned before job j or vice versa, provided the two jobs are scheduled on the same machine.

$$\delta_{\iota j} + \delta_{j\iota} + y_{\iota j} = 1 \quad \iota \in J,\ j \in J,\ \iota < j \tag{22}$$

Constraint set (23) represents the transitivity constraints that ensure a linear order between three jobs.

$$\delta_{\iota j} + \delta_{jk} + \delta_{k\iota} \leqslant 2 \quad \iota \in J,\ j \in J,\ k \in J,\ \iota < j < k \tag{23}$$

Next, constraint set (24) properly calculates variable $y_{\iota j}$.

$$z_{\iota i} + z_{ji} + y_{\iota j} \leqslant 2 \quad \iota \in J,\ j \in J,\ \iota < j,\ i \in \mu \tag{24}$$

Constraint (25) ensures that each job is positioned on a machine.

$$\sum_i z_{\iota i} = 1 \quad \iota \in J \tag{25}$$

Finally, job completion time (regardless of its position) is given by constraint sets (26) and (27).

$$C_j \geqslant p^i_j z_{ji} \quad j \in J,\ i \in \mu \tag{26}$$

$$C_j \geqslant C_\iota + p^i_j(\delta_{\iota j} + z_{\iota i} + z_{ji} - 2) - M(1 - \delta_{\iota j}) \quad \iota \in J,\ j \in J,\ i \in \mu \tag{27}$$

3.5.2. Model M4 for different objective functions

Total weighted completion time: $\min \sum_{j\in J} w_j C_j$, subject to (22)–(27)
Total weighted tardiness: $\min \sum_{j\in J} w_j T_j$, subject to (4), and (22)–(27)
Maximum lateness: $\min L_{\max}$, subject to (5), and (22)–(27)
Total number of tardy jobs: $\min \sum_{j\in J} U_j$, subject to (13), and (22)–(27)

3.5.3. Incorporating ready times in model M4

Job release dates are incorporated into model M4 by adding the following constraint set:

$$C_j \geqslant r_j + p^i_j z_{ji} \quad j \in J,\ i \in \mu \tag{28}$$

4. Experimental study—identical parallel machine environment

The goal of this paper is to identify promising optimization formulation paradigms for various types of parallel machine and objective function environments that can subsequently be used to either (1) solve larger practical scheduling problems of interest to optimality and/or (2) be used to establish tighter lower solution bounds for large scheduling problems under study. Towards this goal, we now focus on the two and three identical parallel machine scheduling problems. We generate a variety of test problems using appropriately specified probability distributions to assess the performance of the proposed MIP modeling paradigms for cases with and without job ready time for both the total weighted completion time ($P2|r_j|\sum w_jC_j$, $P2\|\sum w_jC_j$, $P3|r_j|\sum w_jC_j$, $P3\|\sum w_jC_j$) and makespan objective functions ($P2|r_j|C\max$, $P2\|C\max$, $P3|r_j|C\max$, $P3\|C\max$) individually.

In the experimental cases to be investigated, job processing time p_j is sampled from a discrete uniform (DU) distribution over the range $[1, p_{max}]$ where $p_{max} \in \{20, 100\}$. Each job's weight $w_j \sim DU[1, 10]$ and job release date $r_j \sim DU\left[0, \frac{\alpha}{2}\sum p_j\right]$ with $\alpha \in \{0, 1\}$. We investigate $n \in \{20, 50, 100\}$ jobs in each environment. Finally, 10 replications are generated and analyzed for each of the 2(1)2(3) = 12 factor combinations. Therefore, a total of 12(10) = 120 problem instances are generated for each two and three identical machine environments. Each of the MIP modeling paradigms are modeled in AMPL and analyzed for a maximum of 1 h by CPLEX v10.1's MIP solver using default CPLEX settings on a Windows-based PC with a 3.4 GHz microprocessor and 2 GB of RAM.

4.1. Measures of MIP model efficacy

We use a number of measures to assess each MIP modeling paradigm's efficacy. These measures include (1) the number of test instances solved to optimality within the 1 h time limit, (2) the average solution time for these optimally solved instances, (3) the objective function value of each formulation paradigm's linear programming (LP) relaxation, (4) the number of branch and bound nodes analyzed within 1 h, and (5) the average test instance optimality gap for the test instances which could not be solved in 1 h. These performance measures are also studied by Keha et al. (2009) in their single machine scheduling problem experiments.

We expect each formulation paradigm to produce different levels of solution quality as n, m and p_{max} increase. This should be especially true for the number of jobs n, as this adds complexity to all formulation paradigms. Further, it is expected that the different formulation paradigms will also experience different performance results when job ready/release dates are added. As a point of clarification, all experiments conducted with model M2 include the addition of the valid inequalities.

4.2. Experimental results

4.2.1. Total weighted completion time results

For two identical machines case, the experimental results obtained for the total weighted completion time (TWC) experiments both with and without release dates are given in Tables 1 and 2, respectively. Observing these output tables, the following comments and results become evident. Model M1 is the only formulation paradigm that CPLEX is capable of producing optimal solutions in the TWC experiments with two identical machines case—CPLEX produces feasible solutions for other formulation paradigms at best. The main disadvantage with model M1 is that its LP relaxation is much harder to solve as compared to the other models when the processing time increases. For example, model M1 is not able to solve the LP relaxation for the test cases with 100 jobs and a

Table 1
Results for parallel machine total weighted completion time $P2|r_j|\sum w_jC_j$.

Formulation	N	# of test cases solved to optimality (avg. solution time in seconds)		# of test cases with some integer solution (avg optimality gap)		Avg. # of nodes	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	10 (1)	6 (237)	0	0	1	3346
Network variables (M2)		0	0	10 (7.13%)	10 (8.29%)	278,106	245,989
Assignment and positional date variables (M3)		0	0	10 (94.77%)	10 (95.92%)	467,793	492,569
Linear ordering variables (M4)		0	0	10 (6.41%)	10 (8.7%)	1,294,355	1,364,661
Time indexed variables (M1)	50	0	0	10 (0.03%)	10 (15.05%)	15,337	568
Network variables (M2)		0	0	10 (10.25%)	10 (10.10%)	28,972	26,047
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (100.00%)	44,687	48,630
Linear ordering variables (M4)		0	0	10 (9.63%)	10 (9.51%)	12,188	11,772
Time indexed variables (M1)	100	0	0	10 (0.19%)	0	6194	–
Network variables (M2)		0	0	10 (21.27%)	10 (17.74%)	5033	6006
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (100.00%)	110	88
Linear ordering variables (M4)		0	0	0	0	37	63

Table 2
Results for parallel machine total weighted completion time $P2||\sum w_jC_j$.

Formulation	N	# of test cases solved to optimality (avg solution time in seconds)		# of test cases with some integer solution (avg optimality gap)		Avg # of nodes	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	10 (1)	10 (38)	0	0	0	0
Network variables (M2)		0	0	10 (68.82%)	10 (68.68%)	243,337	240,275
Assignment and positional date variables (M3)		0	0	10 (94.77%)	10 (95.92%)	1,226,793	1,158,985
Linear ordering variables (M4)		0	0	10 (70.18%)	10 (69.72%)	413,909	527,433
Time indexed variables (M1)	50	10 (42)	9 (2173)	0	1 (8.68%)	0	0
Network variables (M2)		0	0	10 (89.05%)	10 (90.82%)	30,711	21,097
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (100.00%)	89,133	84,048
Linear ordering variables (M4)		0	0	3 (89.30%)	2 (89.90%)	6374	6789
Time indexed variables (M1)	100	10 (689)	0	0	0	2	–
Network variables (M2)		0	0	10 (95.35%)	9 (96.28%)	1696	763
Assignment and positional date variables (M3)		0	0	0	0	157	165
Linear ordering variables (M4)		0	0	0	0	56	36

maximum processing time of 100 units. However, M1 yields the best lower bounds among competing formulation paradigms when it can solve the LP relaxation. In terms of the number of branch and bound nodes explored, model M1 explores the least number of nodes in all cases. When job release dates are not present (Table 2), M1 produces a greater number of optimal solutions than in the cases with non-zero release dates. Even in the presence of release date, model M1 still produces the best lower bound estimates among all the competing models.

As model M2's valid inequalities are formulated based on job release dates, model M2 produces much smaller optimality gaps (i.e., better lower bound estimates) in the cases where job release dates exist (Table 1). In the $n = 100$ case with job release dates, model M2 produces the best lower bounds of all competing formulations. In addition, the LP relaxation of model M2 is more tractable/more readily solved than the other formulations. M2 is the only formulation that CPLEX is able to produce a lower bound for *all* test instances. Although models M2 and M4 produce similar lower bounds in cases where CPLEX produces a feasible solution for M4, model M2 explores less branch and bound nodes than M4 in the smaller number of jobs problem instances.

While model M3 is able to produce a lower bound estimate in most experimental instances, its associated optimality gap is significantly higher than the other competing formulation paradigms. Although model M3 explores more branch and bound nodes when compared to the other models when the number of jobs is small and job release dates are not present, it does not appear to be a promising formulation strategy for the TWC problems under study with the current form.

Finally, although model M4 performs comparably to model M2, CPLEX is not able to produce feasible solutions for all tests cases when the number of jobs is increased. Model M4 explores more branch and bound nodes than other formulations when the number of jobs is small and release dates are present. However, when n is increased, the number of nodes explored decreases dramatically. When release dates are added, model M4 produces slightly better lower bounds than does model M2. However, CPLEX does not produce any feasible solution for M4 when the number of jobs increases.

For three identical machines case, the experimental results obtained for the total weighted completion time (TWC) experiments both with and without release dates are given in Tables 3 and 4, respectively. The overall performance of the models with three machine case is superior to the two-machine case. Model M1 and M4 are the formulation paradigms capable of producing optimal solutions in the TWC experiments in three machines case. In compare to the machine environment with two identical machines case, M1 produces more optimal solutions in the case of three machines environment for both zero and non-zero release dates. However, M1 faces the similar difficulties when it comes to increasing n and p_{max}. Notice that M1 is still not able to solve LP relations in the cases $n = 100$ and $p_{max} = 100$ and explores the least number of branch and bound nodes in 1 h time limit.

In three identical machines case, the performance of model M2 for $n = 20$ and $n = 50$ is superior to the case with two machines as it produces better lower bounds. CPLEX is able to produce a feasible solution for model M2 and M3 in all machine environment cases as well. In three identical machines case, comparing M2 and M3 in

Table 3
Results for parallel machine total weighted completion time $P3|r_j|\sum w_jC_j$.

Formulation	N	# of test cases solved to optimality (avg solution time in seconds)		# of test cases with some integer solution (avg optimality gap)		Avg # of nodes	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	10 (1)	10 (24)	0	0	4	9
Network variables (M2)		0	0	10 (5.73%)	10 (7.14%)	87,631	87,247
Assignment and positional date variables (M3)		0	0	10 (91.35%)	10 (92.85%)	731,691	627,821
Linear ordering variables (M4)		1 (3284)	0	9 (1.15%)	10 (1.64%)	1,012,942	1,169,024
Time indexed variables (M1)	50	10 (6)	2 (128)	0	0	0	0
Network variables (M2)		0	0	10 (5.31%)	10 (4.60%)	11,821	8787
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (100.00%)	39,009	46,178
Linear ordering variables (M4)		0	0	10 (1.85%)	10 (2.14%)	39,976	42,117
Time indexed variables (M1)	100	9 (381)	0	1 (0.01%)	0	736	–
Network variables (M2)		0	0	10 (45.16%)	10 (49.48%)	861	879
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (100.00%)	24	38
Linear ordering variables (M4)		0	0	1 (7.00%)	3 (11.14%)	145	748

Table 4
Results for parallel machine total weighted completion time $P3||\sum w_jC_j$.

Formulation	N	# of test cases solved to optimality (avg solution time in seconds)		# of test cases with some integer solution (avg optimality gap)		Avg # of nodes	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	10 (1)	10 (34)	0	0	0	0
Network variables (M2)		0	0	10 (71.94%)	10 (72.85%)	147,734	150,730
Assignment and positional date variables (M3)		0	0	10 (99.18%)	10 (96.48%)	1,487,495	1,491,230
Linear ordering variables (M4)		0	0	10 (59.05%)	10 (58.71%)	583,099	797,847
Time indexed variables (M1)	50	10 (43)	3 (1267)	0	0	0	0
Network variables (M2)		0	0	10 (91.37%)	10 (93.05%)	11,592	10,010
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (100.00%)	108,129	112,171
Linear ordering variables (M4)		0	0	10 (84.61%)	10 (83.61%)	10,870	18,974
Time indexed variables (M1)	100	10 (707)	0	0	0	2	–
Network variables (M2)		0	0	8 (96.45%)	9 (96.83%)	283	33
Assignment and positional date variables (M3)		0	0	0	0	108	115
Linear ordering variables (M4)		0	0	0	0	0	2

terms of optimality gaps, number of nodes explored and the value of initial LP relaxation quality yields similar results as in two identical machines case. Among other formulation paradigms, model M4 is the one that begins to give the best increased performance results when the number of machines is increased to three. In three identical machines case with non-zero release dates, it gives the best lower bounds in the case $n = 50$, 100 and $p_{max} = 100$.

4.2.2. Makespan results

For two identical machines case, the experimental results obtained for the makespan experiments both with and without release dates are given in Tables 5 and 6, respectively. All models produce at least one optimal solution for the $n = 20$ test cases. From the results tables, it appears that models M2, M3, and M4 do not produce optimal solutions as frequently as model M1 does. While model solution efficiency increases for most models when job release dates are present (Table 6), results quality degrades sharply for the test cases with no release dates. As was the case previously for TWC, CPLEX does not produce even feasible solutions for the majority of the models when the number of jobs and maximum job processing time are increased.

Model M1 gives good lower bound estimates for test cases with small number of jobs and small maximum processing time. However,

Table 5
Results for parallel machine maximum completion time $P2|r_j|C_{\max}$.

Formulation	N	# of test cases solved to optimality (avg solution time in seconds)		# of test cases with some integer solution (avg optimality gap)		Avg # of nodes	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	10 (29)	2 (124)	0	8 (7.36%)	429	16,884
Network variables (M2)		1 (529)	0	9 (14.11%)	10 (15.92%)	853,142	637,504
Assignment and positional date variables (M3)		3 (1017)	0	7 (29.21%)	10 (36.52%)	334,915	512,942
Linear ordering variables (M4)		3 (370)	2 (420)	7 (5.00%)	8 (5.45%)	966,274	991,029
Time indexed variables (M1)	50	0	0	10 (11.02%)	2 (46.12%)	19,813	2
Network variables (M2)		0	0	10 (34.39%)	10 (28.79%)	194,206	190,297
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (96.65%)	92,392	98,998
Linear ordering variables (M4)		0	0	10 (16.04%)	10 (17.17%)	34,725	15,110
Time indexed variables (M1)	100	0	0	10 (52.77%)	0	15	–
Network variables (M2)		0	0	10 (36.55%)	10 (36.04%)	20,469	22,523
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (100.00%)	151	190
Linear ordering variables (M4)		0	0	0	0	19	12

Table 6
Results for parallel machine maximum completion time $P2\|C\max$.

Formulation	N	# of test cases solved to optimality (avg solution time in seconds)		# of test cases with some integer solution (avg optimality gap)		Avg # of nodes	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	10 (346)	0	0	10 (45.01%)	547	341
Network variables (M2)		0	0	10 (80.02%)	10 (79.87%)	175,302	247,960
Assignment and positional date variables (M3)		0	0	10 (88.70%)	10 (90.61%)	1,760,225	1,973,788
Linear ordering variables (M4)		0	0	10 (83.06%)	10 (80.70%)	772,598	654,053
Time indexed variables (M1)	50	0	0	10 (50.02%)	0	1135	0
Network variables (M2)		0	0	10 (92.30%)	10 (92.09%)	47,581	75,124
Assignment and positional date variables (M3)		0	0	10 (99.92%)	10 (100.00%)	81,930	151,627
Linear ordering variables (M4)		0	0	10 (93.71%)	10 (92.54%)	5367	8433
Time indexed variables (M1)	100	0	0	6 (74.80%)	0	334	–
Network variables (M2)		0	0	10 (96.46%)	10 (96.59%)	3607	590
Assignment and positional date variables (M3)		0	0	0	0	67	174
Linear ordering variables (M4)		0	0	0	0	0	0

it is clear that better lower bound performance is produced by model M4 than in M1 as the number of jobs and maximum processing time increase. Model M1 is not able to solve the LP relaxation for test cases with $n = 100$ jobs and maximum processing time $p_{max} = 100$. Adding job release dates into the test cases (Table 5) improves model M1's performance more advantageously as compared to the other models performance improvement. Finally, model M1 explores the least number of branch and bound nodes among all competing formulations in all test cases but one (i.e., 100 jobs, no release dates). Model M2 is the only formulation paradigm that CPLEX is able to produce integer solutions for all test cases. In terms of lower bound performance, the performance of model M2 increases (as expected) when job release dates are added. For the test instances with release dates, model M2 provides the best lower bounds in the $n = 100$ job cases while exploring the largest number of branch and bound nodes.

Model M3 is able to provide a feasible solution for most test cases. However, its optimality gap remains between 80% and 100% (i.e., not very tight lower bounds). CPLEX is not able to find a feasible solution for M3 in 1 h for the test cases with $n = 100$ jobs when release dates are omitted. The number of branch and bound nodes explored with model M3 drops as the number of jobs is increased, as it becomes harder to solve the LP relaxation of this model. Model M4 is able to provide optimal solutions more frequently than models M2 and M3. Further, in the case where release dates are non-zero, model M4 produces much tighter lower bounds compared to the other competing models for the test cases with $n = 20$ and $n = 50$ jobs with $p_{max} = 100$. However, model M4 is not able to provide a feasible solution in 1 h for the test cases with $n = 100$ jobs.

For three identical machines case, the experimental results obtained for the makespan experiments both with and without release dates are given in Tables 7 and 8, respectively. The overall performance of the models with three machines case is superior to the two-machine case both in terms of producing more optimal solutions and giving better lower bounds. All models are capable of producing optimal solutions in the makespan experiments for some test instances with three machines and non-zero release dates case. Among other formulation paradigms, model M1 is the one that gives the best increased performance results when the number of machines is increased to three. M1 consistently gives the best lower bounds for all cases. However, increasing the number of jobs and the maximum processing times degrade the performance of model M1, leaving unsolved LP relaxations for the case $n = 100$ and $p_{max} = 100$.

Model M4 has the second most increased performance results after model M1 when the number of identical machines is increased to three. However, its performance sharply decreases in the case $n = 100$ although CPLEX is still able to produce some feasible solutions for some test instances. In three identical machines case, models M2 and M3 give similar performance results in terms of optimality gaps, number of nodes explored and the value of initial LP relaxation quality as in two identical machines case. Notice that CPLEX consistently produces at least a feasible solution for model M2 for all test problem instances in 1 h time limit.

Based on performance improvements observed through the experiments for a three parallel machines case, we carry out additional experiments for both the total weighted completion time and makespan objective functions by increasing the number of

Table 7
Results for parallel machine maximum completion time $P3|r_j|C_{\max}$.

Formulation	N	# of test cases solved to optimality (avg solution time in seconds)		# of test cases with some integer solution (avg optimality gap)		Avg # of nodes	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	10 (1)	10 (349)	0	0 (0.31%)	4	3853
Network variables (M2)		2 (2886)	3 (2593)	8 (6.63%)	7 (8.99%)	612,104	311,027
Assignment and positional date variables (M3)		6 (1935)	6 (1639)	4 (8.77%)	4 (4.52%)	296,714	414,412
Linear ordering variables (M4)		9 (483)	7 (1479)	1 (0.26%)	3 (0.86%)	91,423	531,366
Time indexed variables (M1)	50	5 (1823)	0	5 (1.23%)	0	15,488	0
Network variables (M2)		0	0	10 (20.40%)	10 (21.55%)	50,025	49,934
Assignment and positional date variables (M3)		0	0	10 (99.71%)	10 (98.58%)	48,537	42,914
Linear ordering variables (M4)		0	0	10 (20.19%)	10 (20.38%)	8151	8222
Time indexed variables (M1)	100	4 (2524)	0	6 (30.86%)	0	513	–
Network variables (M2)		0	0	10 (38.48%)	10 (40.62%)	74	81
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (100.00%)	52	45
Linear ordering variables (M4)		0	0	1 (46.99%)	1 (63.22%)	79	109

Table 8
Results for parallel machine maximum completion time $P3\|C\max$.

Formulation	N	# of test cases solv optimality (avg solution time in seconds)		# of test cases with some integer solution (avg optimality gap)		Avg # of nodes	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	10 (225)	0	0	10 (41.17%)	550	126
Network variables (M2)		0	0	10 (71.11%)	10 (73.55%)	180,969	185,004
Assignment and positional date variables (M3)		0	0	10 (87.03%)	10 (88.37%)	1,431,147	1,790,910
Linear ordering variables (M4)		0	0	10 (72.29%)	10 (72.33%)	609,284	601,050
Time indexed variables (M1)	50	0	0	10 (61.15%)	0	162	0
Network variables (M2)		0	0	10 (90.49%)	10 (88.12%)	20,648	30,026
Assignment and positional date variables (M3)		0	0	10 (100.00%)	10 (100.00%)	88,318	138,125
Linear ordering variables (M4)		0	0	10 (90.48%)	10 (89.02%)	6801	4650
Time indexed variables (M1)	100	0	0	7 (82.92%)	0	0	–
Network variables (M2)		0	0	10 (96.85%)	10 (97.01%)	34	33
Assignment and positional date variables (M3)		0	0	0	0	131	180
Linear ordering variables (M4)		0	0	7 (97.60%)	4 (97.54%)	58	30

machines to 5, 10, and 15. In this respect, we generated the same data sets that were previously used for both the two and three parallel machines cases. Each of the MIP modeling paradigms was tested for one replication of each factor combination mentioned earlier. Therefore, 12 instances were generated for each of 5, 10, and 15 identical machine environments. The same measures of MIP model efficacy were monitored during the experiments. The experiment results are as follows.

4.2.2.1. Total weighted completion time results. For the 5, 10, and 15 identical machines cases, the experimental results obtained for the total weighted completion time (TWC) experiments both with and without release dates are given in Tables 9 and 10, respectively. The overall results indicate that the MIP modeling paradigms exhibit different behaviors when considering more parallel machines. The results show that the performance of the models is more or less related to the ratio of the number of jobs to the number of machines (i.e., n/m). The best overall performance is observed by M1. Model M1 continues to be the only formulation paradigm that CPLEX is capable of producing optimal solutions in the TWC experiments for five, ten, and fifteen identical machines cases—CPLEX produces feasible solutions for other formulation paradigms at best. Model M1 is able to solve all the test instances except the cases where the LP relaxations cannot be solved in a 1-h time limit. Increasing the number of machines also sharply decreases the number of B&B nodes explored. Test instances turn out to be solved at the root node. However, for a larger number of jobs, solving LP relaxation becomes much harder for M1 when the maximum processing time increases and the ratio of n/m decreases. Model M1 is not able to solve the LP relaxation for the test cases with 50 and 100 jobs and a maximum processing time of 100 units.

The second best performance results are delivered by model M4 in terms of yielding best lower bounds to the majority of the test instances in the cases with non-zero release dates. The best overall performance improvement can be well observed via M4 when the ratio of n/m decreases. CPLEX is able to give a feasible solution in the 1-h time limit for the cases for which no feasible solution was produced for the counterparts of two and three identical parallel machines. In addition, among other competing formulation paradigms, model M4 explores the largest number of nodes for the majority of test instances.

The additional experimental results with more identical machines involved indicate that model M2 and M3 exhibit opposite behavior as compared to M1 and M4. That is, the performance of M2 and M3 degrades by yielding higher optimality gaps when the ratio of n/m decreases. Although it was possible to have a feasible solution for the cases with two and three identical parallel machines, CPLEX does not produce a feasible solution for M2 when the number of machines is increased to 5, 10, or 15 for 100 jobs. In addition, the number of B&B nodes explored decreases sharply when the ratio of n/m decreases since solving LP relaxation becomes much harder.

4.2.2.2. Makespan results. For the 5, 10, and 15 identical machines cases, the experimental results obtained makespan experiments both with and without release dates are given in Tables 11 and 12, respectively. The MIP modeling paradigms give different results with changes in the ratio of n/m. All models except M3 are capable of producing optimal solutions in the makespan experiments for some test instances with a non-zero release dates case. However, increasing the number of machines degrades the performance of all models for the case $n = 100$. CPLEX is not able to produce a feasible solution for almost all the models for the case $n = 100$. Model M1 gives the best lower bounds for the cases where CPLEX is able to solve the LP relaxations. When the ratio of n/m decreases, the performance of M1 improves for small numbers of jobs.

Model M4 is able to solve all the test instances for the cases with non-zero release dates and $n = 20$ and 50. The best overall performance improvement is demonstrated by M4 when the ratio of n/m decreases except for the case $n = 100$. The performance of model M2 and M3 degrades by yielding higher optimality gaps when the ratio of n/m decreases. The number of B&B nodes explored in 1 h decreases sharply for M2. However, model M3 is able to explore the largest number of B&B nodes for the cases with $n = 20$ and 50.

4.3. Discussion

4.3.1. Linear programming relaxation tractability

Linear programming (LP) relaxations play a very important role in solving mixed integer programming problems. LP relaxations are of interest in branch and bound (B&B) algorithms which partition the whole solution space (branching) and develop lower bounds for portions of the solution space (bounding). Although there are a number of bounding techniques, LP relaxation is one of the best known methods used for providing a lower bound for MIPs. In fact, LP relaxation naturally provides a lower bound since the optimal solution for the "relaxed" problem is always worse than (or equal to) the integer optimal solution.

Tables 13 and 14 illustrate the average root bound percentages for the TWC and makespan problem instances, respectively. Root bound percentages are computed as the ratio of the initial LP relaxation solution to the optimal solution value (or best feasible solution found by any of the competing formulation paradigms). Therefore, a ratio of 1.0 is the best possible outcome. The dashed cells in these two tables refer to the test cases where the LP relaxations could not be solved. Further, we use N/A in cells for the cases

Table 9
Results for additional parallel machine total weighted completion time $Pm|r_j|\sum w_jC_j$.

Formulation	N	Solution time in seconds						Optimality gap						# of nodes					
		$m = 5$		$m = 10$		$m = 15$		$m = 5$		$m = 10$		$m = 15$		$m = 5$		$m = 10$		$m = 15$	
		$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$
Time indexed variables (M1)	20	1	41	2	5	3	132	0	0	0	0	0	0	0	0	0	0	0	0
Network variables (M2)		–	–	–	–	–	–	7.86	9.81	9.08	9.17	12.61	10.27	264	52,000	34,402	30,002	10,992	7743
Assignment and positional date variables (M3)		–	–	–	–	–	–	94.24	87.51	91.98	88.20	93.53	96.27	725,401	1,173,801	747,901	857,001	549,101	303,301
Linear ordering variables (M4)		–	–	–	–	–	–	1.47	1.75	2.32	2.93	5.23	3.35	1,619,301	1,324,301	971,601	988,701	651,501	541,801
Time indexed variables (M1)	50	5	–	10	–	13	–	0	–	0	–	0	–	0	–	0	–	0	–
Network Variables (M2)		–	–	–	–	–	–	8.07	5.42	9.52	5.35	10.01	13.55	14,300	8500	3993	8634	5301	4620
Assignment and positional date variables (M3)		–	–	–	–	–	–	99.88	100	100	100	100	100	140,045	7829	10,275	8894	4694	1720
Linear ordering variables (M4)		–	–	–	–	–	–	2.54	2.86	4.19	3.37	4.61	4.11	302,351	147,701	104,101	100,101	135,201	50,901
Time indexed variables (M1)	100	34	–	55	–	66	–	0	–	0	–	0	–	0	–	0	–	0	–
Network variables (M2)		–	–	–	–	–	–	∞	∞	∞	∞	∞	∞	1068	1083	266	1699	780	320
Assignment and positional date variables (M3)		–	–	–	–	–	–	100	100	100	100	100	100	9	49	0	4	0	0
Linear ordering variables (M4)		–	–	–	–	–	–	9.02	5.21	3.75	2.82	3.09	3.12	11,374	11,248	11,301	23,901	11,501	8701

Table 10
Results for additional parallel machine total weighted completion time $Pm\|\sum w_jC_j$.

Formulation	N	Solution time in seconds						Optimality gap						# of nodes					
		$m = 5$		$m = 10$		$m = 15$		$m = 5$		$m = 10$		$m = 15$		$m = 5$		$m = 10$		$m = 15$	
		$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$
Time indexed variables (M1)	20	1	46	1	29	1	124	0	0	0	0	0	0	0	0	0	0	0	0
Network Variables (M2)		–	–	–	–	–	–	75.59	75.01	76.93	78.74	83.08	83.17	134,208	97,500	30,806	21,903	7481	5383
Assignment and positional date variables (M3)		–	–	–	–	–	–	98.01	95.50	92.60	98.46	99.30	100	2,580,842	1,195,818	1,253,801	1,303,321	815,713	479,390
Linear ordering variables (M4)		–	–	–	–	–	–	44.43	40.92	19.77	25.86	19.66	23.06	1,803,701	1,353,977	994,762	1,052,301	691,501	591,811
Time indexed variables (M1)	50	19	–	76	–	46	–	0	–	0	–	0	–	0	–	0	–	0	–
Network variables (M2)		–	–	–	–	–	–	94.47	94.80	95.42	94.68	96.02	96.19	11,585	3600	90	168	54	0
Assignment and positional date variables (M3)		–	–	–	–	–	–	100	100	100	100	100	100	258,487	153,946	92,101	89,911	86,898	19,314
Linear ordering variables (M4)		–	–	–	–	–	–	82.01	84.35	78.80	77.13	79.76	83.85	301,496	187,956	134,300	132,259	153,600	66,701
Time indexed variables (M1)	100	3300	–	1413	–	934	–	0	–	0	–	0	–	0	–	0	–	0	–
Network variables (M2)		–	–	–	–	–	–	∞	∞	∞	∞	∞	∞	0	0	0	0	0	0
Assignment and positional date variables (M3)		–	–	–	–	–	–	∞	∞	∞	∞	∞	∞	169	197	785	911	555	2001
Linear ordering variables (M4)		–	–	–	–	–	–	97.46	96.69	94.31	95.44	95.44	95.23	0	696	1346	25,701	3679	19,301

Table 11
Results for additional parallel machines maximum completion time $Pm|r_j|Cmax$.

Formulation	N	Solution time in seconds						Optimality gap						# of nodes					
		$m = 5$		$m = 10$		$m = 15$		$m = 5$		$m = 10$		$m = 15$		$m = 5$		$m = 10$		$m = 15$	
		$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$
Time indexed variables (M1)	20	1	44	2	87	4	149	0	0	0	0	0	0	0	0	0	0	0	0
Network variables (M2)		1	–	2260	2	325	63	0	4.82	0	0	0	0	0	225,108	33,470	0	164	0
Assignment and positional date variables (M3)		–	–	–	–	–	–	0.09	5.73	0.16	21.33	65.57	100	486,108	382,179	1,029,701	878,118	624,201	582,379
Linear ordering variables (M4)		4	4	4	5	9	15	0	0	0	0	0	0	125	110	20	80	130	200
Time indexed variables (M1)	50	19	–	35	–	26	–	0	–	0	–	0	–	10	–	7	–	0	–
Network variables (M2)		–	–	–	–	–	–	0.66	12.95	0.08	19.69	12.96	∞	8582	3015	50	111	38	0
Assignment and positional date variables (M3)		–	–	–	–	–	–	80.26	100	100	100	100	100	103,141	43,341	13,601	17,640	7508	880
Linear ordering variables (M4)		109	589	290	443	247	654	0	0	0	0	0	0	246	1981	235	800	626	650
Time indexed variables (M1)	100	938	–	85	–	233	–	0	–	0	–	0	–	1420	–	0	–	1	–
Network variables (M2)		–	–	–	–	–	–	∞	∞	∞	∞	∞	∞	7	68	0	47	0	0
Assignment and positional date variables (M3)		–	–	–	–	–	–	100	100	100	100	100	100	101	120	0	0	0	0
Linear ordering variables (M4)		–	–	–	–	–	–	61.40	24.98	∞	∞	∞	∞	58	931	111	101	0	0

where CPLEX cannot produce a feasible solution for the corresponding model within the allowed 1 h time budget.

Clearly, model M1 provides the best bounds among all formulations. Regardless of whether or not job release dates are present, model M1 consistently gives the best bounds for the cases wherein the LP relaxation is solvable. The size of the time indexed model plays a primary role regarding the difficulty in solving model M1's LP relaxation. In most experiments, model M1's initial lower bound estimate is equal to either the proven optimal solution or the best feasible solution. In this sense, model M1 shows promise for use as an approximation algorithm for parallel machine TWC scheduling problems. This approach which is based on lower bounding techniques has already been applied for single machine scheduling (the reader is referred to Sousa (1989), Sousa and Wolsey (1992), Lee and Sherali (1994), Crama and Spieksma (1995), and Van den Akker, Van Hoesel, et al. (1999) and Van den Akker, Hoogeveen, van de Velde (1999)).

In cases where job release date is considered, the root bounds of model M2 are 80–90%. Even though we increase the number of jobs and the maximum processing times in our experiments, model M2 still generates LP relaxation values which are relatively close to the optimal or best feasible values. However, the LP relaxation performance of model M2 degrades in test cases where release dates equal zero. This is mainly due to the fact that both sub-tour elimination constraints and the proposed valid inequalities are based on the existence of non-zero job release dates. Since model M2 is based on network flow variables, it is possible that an adaptation of the lifting constraint approach by Desrochers and Laporte (1991) can be employed in model M2 to strengthen this second best formulation paradigm.

4.3.2. Model size

The size (memory requirements) of a MIP formulation is of interest as it plays a significant role in defining the space required to store and the time required to solve both LP relaxations and the actual MIP model. The formulations developed in this paper result in a polynomial number of model constraints and model variables. We use different decision variable paradigms to create the four MIP formulation paradigms in this paper. In order to analyze the effect of variable selection on formulation size, Table 15 shows the number of constraints and binary variables associated with each formulation paradigm.

Let l, m, and n denote the upper bound of makespan, the number of machines, and the number of jobs, respectively. As seen in Table 11, we describe the size of each formulation paradigm in terms of these three parameters. Given its formulation size of $O(l)$, one might think model M1 would tend to result in a smaller number of constraints and variables as compared to the other methods. Unfortunately, the number of constraints and variables in model M1 can be enormous, even for small instances, if job processing times are relatively large.

This is the primary challenge in dealing with time indexed formulations—it prohibitively increases storage requirements and model solution times, even for a model's LP relaxation. In this sense, model M1 can have the largest formulation size of all the options. In Van den Akker, Van Hoesel, et al. (1999) and Van den Akker, Hoogeveen, et al. (1999), Dantzig-Wolfe decomposition techniques are applied to alleviate the difficulties arising from the problem size associated with time-indexed model formulation for the single machine case. In their experiments, however, results from this decomposition approach are only promising for small instances of the single machine problem.

Although all other formulation paradigms' sizes are $O(n^2)$, model M4 produces many more constraints than model M3 when the number of jobs is large. While models M2 and M3 produce a similar number of constraints, their overall size is smaller than the

Table 12
Results for additional parallel machines maximum completion time $Pm||C\max$.

Formulation	N	Solution time in seconds						Optimality gap						# of nodes					
		$m = 5$		$m = 10$		$m = 15$		$m = 5$		$m = 10$		$m = 15$		$m = 5$		$m = 10$		$m = 15$	
		$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$ (%)	$p_j \sim [1,100]$ (%)	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$	$p_j \sim [1,20]$	$p_j \sim [1,100]$
Time indexed variables (M1)	20	76	–	4	–	3	142	0	33.52	0	7.71	0	0	575	351	866	27,113	0	0
Network variables (M2)		–	–	402	–	48	50	60.82	52.63	0	15.32	0	0	168,442	233,488	41,126	246,247	0	0
Assignment and positional date variables (M3)		–	–	–	–	–	–	83.72	90.11	85	90.29	85.00	62.00	2,572,355	1,800,512	1,071,771	1,788,111	885,235	634,435
Linear ordering variables (M4)		–	–	1543	–	3	62	53.48	45.90	0	8.74	0	0	1,743,301	1,374,701	594,131	1,234,401	0	11,000
Time indexed variables (M1)	50	–	63	–	–	332	–	44.86	0	48.79	–	0	–	1463	0	133	–	0	–
Network variables (M2)		–	–	–	–	–	–	93.96	94.44	95.15	95.88	∞	∞	17,712	140	39	183	12	0
Assignment and positional date variables (M3)		–	–	–	–	–	–	99.03	100	100	100	100	100	310,221	154,501	87,101	96,001	102,755	19,314
Linear ordering variables (M4)		–	–	–	–	–	–	84.85	85.69	69.69	62.11	48.72	43.55	41,700	13,793	30,820	20,371	79,413	66,701
Time indexed variables (M1)	100	–	–	–	–	–	–	89.36	–	∞	–	∞	–	0	–	0	–	0	–
Network variables (M2)		–	–	–	–	–	–	97.27	∞	∞	∞	∞	∞	62	0	0	0	0	0
Assignment and positional date variables (M3)		–	–	–	–	–	–	∞	∞	∞	∞	∞	∞	0	610	1117	738	1192	689
Linear ordering variables (M4)		–	–	–	–	–	–	∞	∞	92.48	∞	∞	∞	547	0	329	0	0	0

Table 13
Average root bound percentages for TWC cases.

Formulation	n	$r_j \sim \left[0, \frac{\sum p_j}{2}\right]$		$r_j = 0$	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	1.00	1.00	1.00	1.00
	50	1.00	0.85	1.00	0.92
	100	1.00	–	1.00	–
Network variables (M2)	20	0.91	0.90	0.22	0.23
	50	0.89	0.87	0.08	0.08
	100	0.88	0.85	0.03	0.03
Assignment and positional date variables (M3)	20	0.00	0.00	0.00	0.00
	50	0.00	0.00	0.00	0.00
	100	0.00	0.00	N/A	N/A
Linear ordering variables (M4)	20	0.86	0.83	0.15	0.15
	50	0.86	0.88	0.08	0.05
	100	0.83	0.01	0.06	0.01

Table 14
Average root bound percentages for makespan cases.

Formulation	n	$r_j \sim \left[0, \frac{\sum p_j}{2}\right]$		$r_j = 0$	
		$p_j \sim U[1,20]$	$p_j \sim U[1,100]$	$p_j \sim U[1,20]$	$p_j \sim U[1,100]$
Time indexed variables (M1)	20	0.94	0.93	0.56	0.57
	50	0.84	0.83	0.53	0.53
	100	0.74	–	0.48	–
Network variables (M2)	20	0.88	0.86	0.09	0.10
	50	0.82	0.81	0.04	0.04
	100	0.73	0.70	0.02	N/A
Assignment and positional date variables (M3)	20	0.00	0.00	0.00	0.00
	50	0.00	0.00	0.00	0.00
	100	0.00	0.00	0.00	N/A
Linear ordering variables (M4)	20	0.88	0.86	0.09	0.09
	50	0.82	0.81	0.04	0.04
	100	0.73	0.70	0.02	N/A

other formulations—the LP relaxation of these two formulations are solved more rapidly than the other approaches. All formulations except model M1 produce a similar number of binary variables. This is not necessarily a disadvantage, however, as a compact formulation may have a weak LP relaxation (e.g., models M2 and M3), while the relaxation of a model containing a large number of variables (such as M1) can be tight (Savelsbergh, 2001; Nemhauser, 2002).

Pursuing strong bounds for the LP relaxation gives rise to the technique known as column generation, which reveals quite favorable computational results for a variety of parallel machine scheduling problems. Van den Akker, Van Hoesel, et al. (1999) and Van den Akker, Hoogeveen, et al. (1999) demonstrated the computational results of applying column generation to the identical parallel machine scheduling problem with the objective of minimizing total weighted completion time. Chan, Muriel, Simchi-Levi (1998) and Chan, Kaminsky, et al. (1995) analyzed the column generation technique for the same problem by emphasizing on the worst-case performance and probabilistic analysis. On the other hand, Chen and Powell (1999a) showed the computational results of applying column generation techniques to the objectives of minimizing total weighted completion time and weighted number of tardy jobs for the identical, non-identical, and unrelated parallel machine environments. Chen and Powell (1999b) also applied the technique to the objective of minimizing total weighted earliness and tardiness for identical parallel machines problem. In addition, the objective of minimizing maximum lateness for identical parallel machine problems was analyzed by Van den Akker, Hoogeveen, and Van Kempen (2006) in the framework of applying column generation techniques. An overview related to applying column generation techniques to machine scheduling problems with a sum type criterion (e.g., total weighted completion time) can be obtained in Van den Akker, Hoogeveen, and van de Velde (2005).

4.3.3. Disjunctive and conjunctive constraints

The constraints of a mathematical program can be classified as either conjunctive or disjunctive. While conjunctive constraints must be satisfied (i.e., the "and" condition), only one constraint within a set of disjunction constraints must hold (i.e., the "or" condition). For example, consider model M2. Degree constraint sets (7)–(11) are conjunctive constraints that must all be satisfied, while the sub-tour elimination constraint set (12) contains disjunctive constraints of which one must be satisfied.

Although disjunctive constraints are useful in many contexts, such as in linearizing previously non-linear constraints, their primary drawback is their impact on LP relaxations. Models M2–M4 require sufficiently large big-M values in order to result in valid MIP formulations. Unfortunately, tighter LP relaxations result when smaller (if not the feasibly smallest) big-M values are used, assuming that a big-M is required in the formulation paradigm. In this paper, a number of approaches were used to set the value of parameter big-M as small as possible. In addition, valid inequalities (14) and (15) are added to M2 in order to further reduce the solution space, given the fact that big-M constraints exist.

5. Conclusions and future work

In this paper, four different MIP formulation paradigms based on four different types of decision variables are presented for Pm-, Qm-, and Rm-type parallel machine scheduling problems for total weighted completion time, makespan, maximum lateness, total weighted tardiness and total number of tardy jobs performance measures. In addition, these models can easily be adapted to most other objective functions of interest, as all formulations contain job completion time variables.

Computational results for the different numbers of identical machines operating in parallel demonstrate the wide range of effectiveness of the formulation paradigms studied in the environments

Table 15
Model size for each formulation paradigm.

Formulation	Number of constraints	Number of binary variables	Formulation size
Time indexed variables (M1)	$m(l+1)+3n$	$(l+1)mn$	$O(l)$
Network variables (M2)	$mn^2+2mn+2m+3n$	$mn(n+1)$	$O(n^2)$
Assignment and positional date variables (M3)	$mn^2+3mn+n$	$mn(n+1)$	$O(n^2)$
Linear ordering variables (M4)	$\sum_{i=2}^{n-1}\frac{(n-i)(n-i+1)}{2}+n^2+mn^2+2mn$	$2n^2+mn-2n$	$O(n^2)$

of both zero and non-zero job release dates. For the test instances solved to optimality, each model's computation time increases exponentially as the number of jobs and processing times increase. The least computation time is delivered by model M1 for almost all optimally solved cases. However, model M1's computation time most suffers from large processing times. Model M4 provides much better computation time as compared to models M2 and M3. Further, the computation times provided by models M2 and M3 are often higher than M1 and M4, while M2 delivers a relatively lower computation time than model M3. At a high level, while experimental results suggest that most paradigms can find the optimal solution to at least a single problem instance, model M1 is able to provide optimal solutions more frequently than the other models. Further, model M1 also yields tight lower bounds for the problem instance cases wherein the optimal solution is not reached within the allowed 1 h time limit. However, it is very hard to solve the LP relaxation of model M1, especially in the cases where the number of jobs and the maximum processing time are increased.

In cases where non-zero job release dates exist, the best performance results are delivered by model M1 in terms of yielding better lower bounds. That is due to the fact that the binary variables are set to zero by constraint (7), which facilitates the branch and bound solution procedure. Model M2 has the second best performance for the cases with non-zero job release dates—this is due in part to the fact that model M2's valid inequalities take advantage of job release dates. The LP relaxation of this formulation is solved much faster than the other competitors. In fact, CPLEX produces feasible solutions for model M2 for cases with which the other models are not able to do so. Since model M2 is based on network flow variables, one potential direction of future research could be to improve the model's sub-tour elimination constraints by employing a lifting approach. Model M3 remains the loosest formulations in our experiments by producing optimality gaps in excess of 90% in most cases. However, it is observed during the experiments that the feasible solutions obtained by model M3 are very close to the optimal or best feasible solutions for most of the test cases. That is due to the fact that the lower bound obtained by the LP relaxation of M3 is so loose that it is not able to converge an optimal solution in 1 h time limit, which is a natural result of disjunctive constraints in the model. A set of valid inequalities could be a proper approach to improve the quality of lower bound obtained by M3. Although model M4 gives good lower bounds for small number of jobs instances, its performance degrades in larger cases.

Based on the experimental findings, it is recommended to use model M1 for *P2* scheduling problems when job processing time is small; otherwise, model M2 is recommended when either job processing times are large and/or non-zero job release dates exist. Additional future work is to explore the possibility of developing valid inequalities for models in order to remove or alleviate the presence of the disjunctive constraints that cause huge optimality gaps. Since the formulations developed in this paper afford the opportunity to conduct experiments over a variety of machine scheduling environments, our additional future work is to compare the formulations for different types of objective functions ($\min \sum_{j\in J} w_j T_j$, $\min L_{\max}$ and $\min \sum_{j\in J} U_j$) in different job processing both identical and non-identical machine environments (*Pm, Qm and Rm*).

References

Balas, E. (1985). On the facial structure of scheduling polyhedra. *Mathematical Programming, 24*, 179–218.

Blazewicz, J., Dror, M., & Weglarz, J. (1991). Mathematical programming formulations for machine scheduling: A survey. *European Journal of Operational Research, 51*, 283–300.

Cakici, E., & Mason, S. J. (2007). Parallel machine scheduling subject to auxiliary resource constraints. *Production Planning and Control, 18*, 217–225.

Chan, L. M. A., Kaminsky, P., & Simchi-Levi, D. (1995). Machine scheduling, linear programming and list scheduling heuristics. Technical Report, Northwestern University, Chicago.

Chan, L. A., Muriel, A., & Simchi-Levi, D. (1998). Machine scheduling, linear programming and list scheduling heuristics. *Operations Research, 46*(5), 729–741.

Chen, Z. L., & Powell, W. B. (1999a). Solving parallel machine scheduling problems by column generation. *Informs Journal on Computing, 11*, 78–94.

Chen, Z. L., & Powell, W. B. (1999b). A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. *European Journal of Operational Research, 116*, 220–232.

Chudak, F. A., & Hochbaum, D. S. (1999). A half- integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters, 25*, 199–204.

Crama, Y., & Spieksma, F. C. R. (1995). Scheduling jobs of equal length: Complexity, facets and computational results. In E. Balas & J. Clausen (Eds.), *Proceedings of the 4th international IPCO conference, Denmark. Lecture notes in computer science* (Vol. 920, pp. 277–291). Berlin: Springer.

Dauzère-Pérès, S. (1997). An efficient formulation for minimizing the number of late jobs in single-machine scheduling. In IEEE symposium on emerging technologies & factory automation (ETFA), pp. 442–445.

Dauzère-Pérès, S., & Sevaux, M. (2003). Using Lagrangean relaxation to minimize the weighted number of late jobs on a single machine. *Naval Research Logistics, 50*(3), 273–288.

Desrochers, M., & Laporte, G. (1991). Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. *Operations Research Letters, 10*, 27–36.

Dyer, M. E., & Wolsey, L. A. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics, 26*, 255–270.

Houck, D. J., & Vemuganti, R. R. (1976). The traveling salesman problem and shortest n-paths, presented at ORSA–TIMS meeting, Philadelphia, PA (May).

Kara, I., Laporte, G., & Bektas, T. (2004). A note on the lifted Miller–Tucker–Zemlin subtour elimination constraints for the capacitated vehicle routing problem. *European Journal of Operational Research, 158*, 793–795.

Keha, A., Khowa la, K., & Fowler, J. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers and Industrial Engineering*, 357–367.

Kulkarni, R. V., & Bhave, P. R. (1985). Integer programming formulations of vehicle routing problems. *European Journal of Operational Research, 20*, 58–67.

Lasserre, J. B., & Queyranne, M. (1992). Generic scheduling polyhedral and a new mixed integer formulation for single-machine scheduling. In Proceedings of the second IPCO conference, Carnegie-Mellon University Pittsburgh, pp. 136–149.

Lee, Y., & Sherali, H. D. (1994). Unrelated machine scheduling with time-window and machine downtime constraints: An application to a naval battle-group problem. *Annals of Operations Research, 50*, 339–365.

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulations and traveling salesman problems. *Journal of the Association for Computing Machinery, 7*, 326–329.

Nemhauser, G. L. (2002). Fourth international workshop on integration of AI and OR techniques in constraint programming for combinatorial optimization problems, Le Croisic, France, pp. 23–24 (March).

Nemhauser, G. L., & Savelsbergh, M. W. P. (1992). A cutting plane algorithm for the single machine scheduling problem with release times. In M. Akgül, H. Hamacher, & S. Tufecki (Eds.), *Combinatorial optimization: New frontiers in the theory and practice. NATO ASI series F: Computer and systems sciences* (Vol. 82, pp. 63–84). Berlin: Springer.

Pinedo, M. (2002). *Scheduling: Theory, algorithms, and systems* (2nd ed.). New Jersey: Prentice-Hall.

Queyranne, M. (1993). Structure of a simple scheduling polyhedron. *Mathematical Programming, 58*, 263–285.

Queyranne, M., & Schulz, A. S. (1994). Polyhedral approaches to machine scheduling. Technical report 408/1994, Department of Mathematics, Technical University of Berlin, Berlin, Germany.

Queyranne, M., & Wang, Y. (1991). Single-machine scheduling polyhedra with precedence constraints. *Mathematics of Operations Research, 16*, 1–20.

Savelsbergh, M. W. P. (2001). Branch-and-price: Integer programming with column generation. *Encyclopedia of Optimization*.

Šoric, K. (2000). A cutting plane algorithm for a single machine scheduling problem. *European Journal of Operational Research, 127*, 383–393.

Sousa, J. P. (1989). Time-indexed formulations of non-preemptive single-machine scheduling problems. Ph.D. Thesis, Catholic University of Louvain, Louvain-la-Neuve.

Sousa, J. P., & Wolsey, L. A. (1992). A time-indexed formulation of non-preemptive single machine scheduling problems. *Mathematical programming, 54*, 353–367.

Van den Akker, J. M., Hoogeveen, J. A., & Van Kempen, J. W. (2006). Parallel machine scheduling through column generation: minimax objective functions (extended abstract). In Y. Azar, & T. Erlebach (Eds.) ESA 2006. LNCS 4168, Springer, pp. 648-659.

Van den Akker, J. M., Hoogeveen, J. A., & van de Velde, S. L. (1999). Parallel machine scheduling by column generation. *Operations Research, 47*, 862–872.

Van den Akker, J. M., Hoogeveen, J. A., & van de Velde, S. L. (2005). *Applying column generation to machine scheduling: Column Generation.* Springer. 303-330.

Van den Akker, J. M., Hurkens, C. A. J., & Savelsbergh, M. W. P. (2000). Time-index formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing, 12*, 111–124.

Van den Akker, J. M., Van Hoesel, C. P. M., & Savelsbergh, M. W. P. (1999). A Polyhedral approach to single machine scheduling problems. *Mathematical Programming, 85*, 541–572.