Universidade Federal de Minas Gerais

Escola de Engenharia

Departamento de Engenharia de Produção

Programa de Pós-Graduação em Engenharia de Produção

Scheduling Problem with Unrelated Parallel Machines: Mathematical Formulation, Decomposition Methods and Hybridization

Francisco Regis Abreu Gomes

regisgomes@ifce.edu.br

Orientador: Geraldo Robson Mateus

mateus@dcc.ufmg.br

Belo Horizonte, Agosto 2017

Universidade Federal de Minas Gerais

Escola de Engenharia

Departamento de Engenharia de Produção

Programa de Pós-Graduação em Engenharia de Produção

Scheduling Problem with Unrelated Parallel Machines: Mathematical Formulation, Decomposition Methods and Hybridization

Tese apresentada ao curso de Pós-Graduação em Engenharia de Produção da Universidade Federal de Minas Gerais como requisito parcial para obtenção do grau de Doutor em Engenharia de Produção.

Área de Concentração: Pesquisa Operacional e Engenharia de Manufatura, linha de pesquisa Modelos e Algoritmos de Otimização para Sistemas em Redes e de Produção.

Orientador: Geraldo Robson Mateus.

Francisco Regis Abreu Gomes

Belo Horizonte, Agosto 2017

Gomes, Francisco Regis Abreu.
G633s Scheduling problem with unrelated parallel machines [manuscrito] : mathematical formulation, decomposition methods and hybridization / . Francisco Regis Abreu Gomes .- 2017.
viii, 62 f., enc.: il.

Orientador: Geraldo Robson Mateus.

Tese (doutorado) Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 16-62.

Bibliografia: f. 12-15.

1. Engenharia de produção - Teses. 2. Método de decomposição - Teses. 3. Máquinas paralelas - Teses. 4. Formulação matemática - Teses. I. Mateus, Geraldo Robson. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 658.5(043)

Universidade Federal de Minas Gerais

Escola de Engenharia

Departamento de Engenharia de Produção

Programa de Pós-Graduação em Engenharia de Produção

FOLHA DE APROVAÇÃO

Scheduling Problem with Unrelated Parallel Machines: Mathematical Formulation, Decomposition Methods and Hybridization

Francisco Regis Abreu Gomes

Tese defendida e aprovada pela banca examinadora constituída por:

Prof. Geraldo Robson Mateus – Orientador
UFMG

Prof. Rafael Castro de Andrade
UFC

Prof. Reinaldo Morabito Neto
UFSCAR

Prof. Martin Gomes Ravetti
UFMG

Prof. Maurício Cardoso de Souza
UFMG

Belo Horizonte, 7 de agosto de 2017

Resumo

Esta tese aborda dois problemas de sequenciamento de máquinas paralelas não-relacionadas com os tempos de preparação dependentes da máquina e da sequência. Ambos os problemas são NP-hard. As diferenças entre esses problemas são a função objetivo adotada e os métodos de solução utilizados. No primeiro problema, o makespan é função objetiva e a decomposição Benders combinatória é o método. Este método pode ser lento para convergir. Portanto, três procedimentos são introduzidos para acelerar sua convergência. O primeiro procedimento consiste em encerrar a execução do problema mestre quando uma solução ótima repetida é encontrada. O segundo procedimento é baseado na técnica multicut. Por fim, o terceiro procedimento é baseado no procedimento warm-start. O esquema de decomposição Benders combinatório melhorado é comparado com uma formulação matemática e uma implementação convencional da decomposição de Benders. Nos experimentos são utilizados dois conjuntos de testes da literatura. Para o primeiro conjunto, o método proposto executa 21,85% em média mais rápido do que a implementação convencional. Para o segundo conjunto, o método proposto não conseguiu encontrar uma solução ótima em apenas 31 em 600 instâncias, obteve uma diferença entre os limites inferiores e superiores de 0,07% e teve um tempo computacional médio de 377,86 s, enquanto os melhores resultados dos outros métodos foram 57, 0,17 % e 573,89 s, respectivamente. No segundo problema o atraso total é a função objetivo. Os modelos matemáticos para este problema em geral usam uma constante conhecida como big-M devido às restrições disjuntivas. Isso produz limites inferiores muito fracos que dificultam a obtenção da solução ótima, mesmo para instâncias de pequeno porte. Para resolver este problema é proposta uma formulação matemática que não use a constante big-M. Para este fim, apresenta-se uma abordagem que usa tarefas fictícias em vez da constante big-M. Além disso, é proposta uma condição de otimização que reduz o espaço de busca da solução do problema. Os experimentos realizados em cinco tipos de instâncias produziram prova computacional da superioridade do modelo proposto em comparação com modelos baseados nas formulações de Wagner (1959) e Manne (1960). O modelo proposto produziu 291 soluções ótimas em comparação com 98 e 148 dos modelos de Wagner (1959) e Manne (1960), respectivamente, e foi até três ordens de magnitude mais rápido nas 300 instâncias de pequeno porte que foram testadas. Um algoritmo de geração de coluna também é proposto para encontrar soluções quase ótimas para instâncias de tamanho médio com até 50 tarefas e dez máquinas. Ao contrário das abordagens padrão, o modelo proposto é usado em vez de um algoritmo de programação dinâmica para resolver o problema de pricing. Para acelerar a convergência do algoritmo de geração de colunas, várias heurísticas são propostas para gerar as colunas iniciais e resolver o problema de pricing. A geração de colunas híbrida obteve uma diferença média entre os limites inferiores e superiores e tempo de execução de 2,71% e 930,48 s, respectivamente, em comparação com 34,78% e 2.490,37 s, respectivamente, em relação ao modelo proposto. Os resultados indicam que as abordagens propostas são mais eficazes em tempo de execução e qualidade da solução.

Palavras-chave: *sequenciamento, máquinas paralelas, decomposição de Benders, formulação matemática, geração de colunas, hibridização.*

Abstract

This thesis addresses two unrelated parallel machines scheduling problem with sequence and machine dependent setup times. Both problems are NP-hard. The differences between these problems are the objective function adopted and the solution methods used. In the first problem the makespan is objective function and combinatorial Benders decomposition is solution method. This method can be slow to converge. Therefore, three procedures are introduced to accelerate its convergence. The first procedure consists of terminating the execution of the master problem when a repeated optimal solution is found. The second procedure is based on the multicut technique. The third procedure is based on the warm-start technique. The improved combinatorial Benders decomposition scheme is compared to a mathematical formulation and a standard implementation of Benders decomposition algorithm. In the experiments, two test sets from the literature are used. For the first set the proposed method performs 21.85% on average faster than the standard implementation of the Benders algorithm. For the second set the proposed method failed to find an optimal solution in only 31 in 600 instances, obtained an average gap of 0.07%, and took an average computational time of 377.86 s, while the best results of the other methods were 57, 0.17%, and 573.89 s, respectively. In the second problem the total tardiness is objective function. Mathematical models for this problem often use a constant known as big-M because the disjunctive constraints. This yields very weak lower bounds that make it difficult to obtain the optimal solution, even for small-size instances. To address this problem is proposed a mathematical formulation that does not use the big-M constant. To this end is presented an approach that uses dummy jobs instead of the big-M constant. Additionally, an optimality condition method that reduces the solution space of the problem is proposed. Experiments conducted on five instance types produced computational proof of the superiority of the proposed model compared to models based on Wagner's (1959) and Manne's (1960) formulations. The proposed model produced 291 optimal solutions compared to 98 and 148 of Wagner's (1959) and Manne's (1960) models, respectively, and it was up to three orders of magnitude faster in the 300 small-size instances that were tested. A column-generation algorithm is also proposed to find near-optimal solutions for medium-size instances with up to 50 jobs and 10 machines. Unlike standard approaches, the proposed model is used instead of a dynamic programming algorithm to solve the pricing problem. For accelerating the convergence of the column-generation algorithm, various heuristics are proposed to generate the initial columns and solve the pricing problem. The hybrid column generation obtained an average gap and runtime of 2.71% and 930.48 s, respectively, compared to 34.78% and 2,490.37 s, respectively, of the proposed model. Results indicate that the proposed approaches are more effective in terms of both running time and solution quality.

Keywords: *scheduling, parallel machines, Benders decomposition, mathematical formulation, column generation, hybridization.*

Acknowledgments

To God for giving me health and strength daily to continue studying and working.

Naming all who collaborated in the development of this thesis is an arduous task. So I thank first all the anonymous people who somehow helped to conclude this journey.

To my advisor, prof. Dr. Geraldo Robson Mateus, for the understanding and patience that led the distance orientation and the knowledge sharing.

A special thanks to my wife, Layse Silveira Braga, for always being of my side, for her love and patience during my absences to produce the thesis.

My mother, Regina Fatima de Abreu Gomes, for having taught me the importance of studies and hard work.

To my great friend, Caio Colares Vasconcelos, for your prayers and encouragement words.

To the Production Engineering Department of UFMG, mainly through professors Rodrigo Ribeiro, Martín Gómez Ravetti, and Ricardo Saraiva de Camargo for institutional support and knowledge sharing.

To the Federal Institute of Education, Science and Technology of Ceará, but specifically to my directors at the time of approval in the postgraduate program, the general director prof. Aristides de Souza Neto and the teaching director prof. Francisco de Assis Rocha da Silva.

Finally, to all who contributed in some way to the accomplishment of this work.

Thank you very much!

Contents

List of Tables

1. Introduction

Scheduling has been a subject of a significant amount of literature in the operations research field since the early 1950s (Gupta and Stafford Jr., 2006). The main objective of scheduling is an efficient allocation of shared resources over time to activities, where jobs represent activities and machines represent resources. Application areas for scheduling theory are manufacturing, computers, transportation, services, etc. Among the scheduling problems, those with parallel machines are quite studied. Parallel machine scheduling problems (PMSP) address the scheduling of n jobs on m machines to minimize some function, which is typically a function of completion time or tardiness of jobs. To learn more about these kinds of problems, the survey produced by Mokotoff (2001) can be consulted. PMSPs are known to be NP-hard (Pinedo, 2012).

1.1 Problem Definition

The problem addressed in this thesis is a PMSP with unrelated machines (UPMSP), where a set N of jobs is scheduled on a set M of machines. Each job j takes processing time p_{ij} on machine i. The system machines are unrelated, which means that job j can have a processing time that is longer than job k on a specific machine, although the same cannot be true for another machine. There is a setup time, s_{ijk}, which corresponds to the time required between the end of job j and the beginning of job k on machine i. Setup times comply with the triangular inequality $s_{ijk} \leq s_{ijl} + s_{ilk}$. In this model, it is necessary to use a dummy job 0, with all its parameters equal to zero. Moreover, it is the first and last jobs of the sequences, where N_0 is the set of jobs N plus dummy job 0. The goal of the problem is to determine a schedule of job assignments for the machines that minimizes the makespan or total tardiness. Using the three-element notation of Graham et al. (1979), this problem can be classified as $R \mid sds \mid C_{max}$ (or ΣT_j).

The following presented mathematical model can be used to measure both performance criteria, makespan or total tardiness, where variables y_{ij} is one if job j is processed in machine i (and 0, otherwise), x_{ijk} is one if the k is processed immediately after job j in machine i (and 0, otherwise), C_j denotes the completion time of job j, and C_{max} is the makespan of the solution, and T_j is the tardiness of job j. The model itself is as follows:

$$\min C_{max} \ \ (or \sum_{j \in N} T_j) \qquad (1)$$

$s.a$:

$$\sum_{i \in M} y_{ij} = 1, \qquad \forall j \in N \qquad (2)$$

$$\sum_{\substack{j \in N_0 \\ j \neq k}} x_{ijk} = y_{ik}, \qquad \forall k \in N, i \in M \qquad (3)$$

$$\sum_{\substack{k\in N_0 \\ j\neq k}} x_{ijk} = y_{ij}, \qquad \forall j \in N, i \in M \qquad (4)$$

$$\sum_{k\in N} x_{i0k} \leq 1, \qquad \forall i \in M \qquad (5)$$

$$C_k \geq C_j + s_{ijk} + p_{ik} - (1 - x_{ijk})M, \qquad \forall j \in N_0, k \in N, j \neq k, i \in M \qquad (6)$$

$$C_{max} \geq C_j, \qquad \forall j \in N \qquad (7)$$

$$or\ T_j \geq C_j - d_j, \qquad \forall j \in N \qquad (7)$$

$$C_0 = 0, \qquad (8)$$

$$C_j \ \geq 0, \qquad \forall j \in N \qquad (9)$$

$$or\ T_j \ \geq 0, \qquad \forall j \in N \qquad (9)$$

$$x_{ijk} \in \{0,1\}, \qquad \forall j, k \in N_0, j \neq k, i \in M \qquad (10)$$

$$y_{ij} \in \{0,1\}, \qquad \forall j \in N, i \in M \qquad (11)$$

Objective (1) minimizes the makespan or total tardiness of the solution. Constraints (2) ensure that each job is processed by only one machine. Constraints (3) and (4) ensure that each job has only one predecessor and successor, respectively. Constraints (5) ensure that a maximum of one job is scheduled as the first job on each machine. Constraints (6) ensure the correct order of the jobs and the sub-cycles elimination, that is, if x_{ijk} = 1 the completion time of job *k* should be greater or equal than the completion time of job *j*, and if x_{ijk} = 0, these constraints become redundant. Constraints (7) define the makespan or the tardiness of each job in the solution. Constraints (8) assign 0 to the completion time of the dummy job. Finally, constraints (9) to (11) define the non-negativity and integrality of the variables.

It is difficult to find an optimal solution for the problem even for small instances. One of the difficulties is based on the big-*M* constant used in constraints (6) to calculate the completion time of jobs.

In the first problem the objective function is makespan and combinatorial Benders decomposition is solution method. This method can be slow to converge. Therefore, three procedures are introduced to accelerate its convergence. The first procedure is a new method that consists of terminating the execution of the master problem when a repeated optimal solution is found. The second procedure is based on the multicut technique. The third procedure is based on the warm-start. The improved Benders decomposition scheme is compared to a mathematical formulation and a standard implementation of Benders decomposition algorithm. The results of this approach are presented in Appendix A. In the second problem the total tardiness is objective function and a mathematical formulation and a column-generation algorithm combined with heuristics are proposed to solve it. Mathematical models for this problem in general use a big-M constant to guarantee the disjunctive constraints. This yields very weak lower bounds that make it difficult to obtain the optimal solution, even for small-size instances. To address this problem is proposed a mathematical formulation that does not use the big-M constant. To this end is presented an approach that uses dummy jobs instead of the big-M constant. Additionally, an optimality condition method that reduces the solution space of the problem is proposed. A column-generation algorithm is also proposed to find near-optimal solutions for medium-size instances with up to 50 jobs and 10 machines. Unlike standard approaches, the proposed model is used instead of a dynamic programming algorithm to solve the pricing problem. For accelerating the convergence of the column-generation algorithm, various heuristics are proposed to generate the initial columns and solve the pricing problem. The results of these approaches are presented in Appendix B.

1.2 Related Works

Among the works with criterion makespan the most use heuristics and metaheuristics to solve the UPMSP, Vallada and Ruiz (2011) use two versions of a genetic algorithm. Fleszar et al. (2012) develop a variable neighborhood descent search (VNDS) algorithm, which is hybridized with mathematical programming elements. Ying et al. (2012) propose a restricted simulated annealing algorithm. Arnout et al. (2014) present an ant colony algorithm with two stages. Finally, Avalos-Rosales et al. (2015) propose three versions of a method based on multi-start and VNDS algorithms. To obtain an optimal solution for this problem, Tran and Beck (2012) present an algorithm based on a logic-based Benders decomposition and Avalos-Rosales et al. (2015) explore many mathematical formulations with a new linearization to calculate the makespan.

Papers with the total tardiness criterion are showed following. Bilge et al. (2004) propose a tabu search where candidate list strategies, tabu classifications, tabu tenure and intensification/diversification strategies are investigated. Anghinolfi and Paolucci (2007) propose a hybrid metaheuristic that incorporates the core features of simulated annealing, tabu search and variable neighborhood search. Armentano and de França Filho (2007) propose a GRASP-based search heuristics that incorporate adaptive memory principles. Lin et al. (2011) present an iterated greedy heuristic. Lee et al. (2013) evaluate an algorithm based on tabu search.

For minimizing the weighted number of tardy jobs, M'Hallah and Bulfin (2005) suggested branch and bound algorithms, Chen and Chen (2009) propose a hybrid metaheuristics that integrate the tabu search and the variable neighborhood descent approach, and Chen (2012) proposes several iterated hybrid metaheuristic algorithms. Chen (2009) considers the problem with an additional strict due date constraint for some jobs, and proposes a simulated annealing algorithm that incorporates the feasibility improvement method, Lin et al. (2011) propose an iterated greedy algorithm and a simple dispatching rule referred as primary customers and the shortest completion time first to generate a initial solution. Lee et al. (2013) propose a tabu search algorithm that incorporates various neighborhood generation methods, and it has performance compared with the two previous methods.

Following are the works that combine more than one performance criterion. Chen and Powell (1999) address several PMSPs considering identical, uniform, and unrelated machines with two objectives: to minimize the total weighted completion time and to minimize the weighted number of tardy jobs. Paula et al. (2007) propose an approach based on variable neighborhood search to minimize the makespan and the sum of weighted tardiness of each job. Rocha et al. (2008) propose a branch-and-bound approach to minimize the makespan and the sum of weighted tardiness of each job. For the UPMSP with machine and sequence-dependent setup time, not many studies have been done on this problem. Paula et al. (2010) propose a non-delayed relax-and-cut algorithm based on the Lagrangian relaxation of a time-indexed formulation to minimize the total weighted tardiness. For minimizing the total earliness and tardiness penalties, Nogueira et al. (2014) propose three different heuristics based on GRASP metaheuristic, and Zeidi and Hosseini (2015) propose a genetic algorithm with simulated annealing method as local search procedure to improve the quality of solutions. Lopes and Carvalho (2007) address the UPMSP with sequence-dependent setup times and availability dates for the machines and release dates for the jobs. The authors failed to identify any structural dominance property, but to overcome this drawback have proposed a new method to reduce the search space to accelerate the solving from a 2-cycle elimination version of a dynamic programming algorithm proposed by authors. Finally they use it in a branch-and-price algorithm to minimize the weighted tardiness.

1.3 Objectives

The exact methods such as branch and bound and dynamic programming techniques, while guarantee an optimal solution, could address only a subset of problems that are small in terms of the number of variables and constraints. Therefore, the majority of studies uses heuristics or metaheuristics to solve larger problems in less computational time. These methods have the disadvantage of not guaranteeing the quality of their solutions, that is, they do not present a dual solution that can be compared with the primal solution found in order to calculate an optimality gap.

Then, the main objective of this thesis is to study methods that combine the advantages of the exact and heuristic methods to solve the UPMSP with makespan or total tardiness as performance criteria. The specific objectives below are also explored in Appendices A and B, respectively.

Considering the makespan criterion

- To propose heuristics to accelerate the convergence of the Benders decomposition algorithm.

- To compare a standard implementation of a Benders decomposition algorithm (Tran and Beck, 2012), the state-of-the-art mathematical model (Avalos-Rosales et al, 2015), and the proposed Benders decomposition algorithm.

Considering the total tardiness criterion

- To develop a mathematical model that does not use the big-M constant to linearize the precedence constraints.

- To compare the proposed mathematical model with two other models from literature.

- To develop a hybrid column generation algorithm to overcome the deficiencies of the proposed mathematical model.

2. Methods

2.1 Linear Relaxation

The tightness of the relaxed feasible region, which is reflected in the lower bound (for minimization problem) is an important criterion when solving the original MILP. Tighter relaxed feasible region reduces the search space of solutions (Lee and Grossmann, 2000). Large values for M can easily lead to numerical problems and weak linear relaxations (Camm, et al., 1990; Klotz and Newmann, 2013). For example, considering two jobs j and 1, to guarantee that the constraints (6) are valid for all j, $k \in N$, the big-M constant must be greater than or equal to the greatest difference between all completion times of jobs. If an appropriate value for the big-M is equal to 1000 and p_{ik} and s_{ijk} are equals to 100, simply x_{ijk} adopting a value less than or equal to 0.8 (very close to the integer value) in the relaxed solution the difference between C_k and C_j would be equal to 0. Therefore, linear relaxations of scheduling problems that use constraints with a big-M constant are so weak. That makes it difficult to obtain an optimal solution even for small instances.

There are formulations for PMSPs that do not use the big-M constant to represent the precedence between jobs, including time-indexed models. In these models, job precedence decisions are represented by discrete variables under a time horizon. The linear relaxation provided of a time-indexed formulation is very stronger than the bounds provided by the linear relaxations of many other MILP formulations (Dyer and Wolsey, 1990). Unfortunately, the promising computational results reported have all been for relatively small instances. Even for relatively small instances the number of constraints and the number of variables can be huge. As a result, the memory required to store an instance and the time required to solve just the linear relaxation may be prohibitive. This brings to the main weakness of time-indexed formulations: their size. (Van Den Akker et al., 2000). This formulation were tested in the problems addressed, but the memory failure appeared quickly even for very small instances, so their results are not presented.

However, Avalos et al. (2015) propose a method to calculate the makespan of a PMSP independent of the big-M constant. The completion time of machine O_i, $i \in M$, is given by the sum of the processing (p_{ijk}) and setup (s_{ijk}) times of all the jobs (x_{ijk}) allocated to the machine i. The makespan is the largest value of O_i among all machines i. The proposal from Avalos et al. (2015) represented by constraints (12) and (13) are showed below. Although these constraints do not eliminate the need of constraints (6) to guarantee precedence between jobs, the results regarding the linear relaxation quality, optimal solutions number and runtime obtained are much better than those presented by the previous models.

$$O_i \geq \sum_{\substack{j \in N_0 \\ j \neq k}} \sum_{k \in N} s_{ijk} x_{ijk} + \sum_{\substack{j \in N_0 \\ j \neq k}} \sum_{k \in N} p_{ij} x_{ijk}, \qquad \forall i \in M \qquad (12)$$

$$C_{max} \geq O_i, \qquad \forall i \in M \qquad (13)$$

Previously Tran and Beck (2012) propose a combinatorial Benders decomposition algorithm for the same problem addressed by Avalos et al. (2015). The results obtained by them are also much better than previous models based on constant big-*M*. This is obtained using a combinatorial relaxation of constraints (12), where sub-cycles in the job sequences may exist because the constraints (6) are not used. But these two approaches have not been compared to each other, so it is proposed to do that.

2.2 Dummy job

An analysis was carried out on the possibility of using the model from Avalos et al. (2015) or the Benders decomposition method to minimize the total tardiness of PMSPs, since they had so good results with the minimization of makespan. But it was verified that these two methods cannot be applied in this case with the same efficiency, because when the criterion is makespan to minimize the order of the jobs in the sequence usually has little impact in the makespan value. While the performance criterion is the total tardiness time of jobs, different job sequences can have a significant effect on the value of tardiness jobs. Therefore, the constraints (12) cannot improve the process of obtaining the total tardiness value of the jobs. To illustrate that consider two jobs *j* and *1*, with processing times equal to 10 and due times (d_j) 4 and 50, respectively. The tardiness of a job *j* (T_j) is calculated as max (0, C_j - d_j) and the total tardiness (*TT*) is a sum of the tardiness of all jobs of the sequence. The following are the results for the two possible solutions.

Table 1 - Fictitious sequences and values of performance criteria

Seq.	C_j	C_l	C_{max}	T_j	T_l	*TT*
j, l	10	20	20	*max* {0, 10 - 4}= 6	*max* {0, 20 - 50}= 0	6
l, j	20	10	20	*max* {0, 20 - 4}= 16	*max* {0, 10 - 50}= 0	16

Inspired by the results achieved by Avalos et al. (2015), a mathematical model for scheduling problems with parallel machines and performance criteria to minimize the total tardiness that does not use the big-*M* constant is developed. This was possible using Wagner's model (1959) variables and a dummy variable. The Wagner's model (1959) uses positional variables x_{ijp} equal to 1 if job *j* is allocated at position *p* of the job sequence in machine *i* and 0, otherwise. The number of positions in a machine is equal to the number of jobs *N*. Since there is a chance that positions will not be occupied by jobs, the model uses the big-*M* constant to handle it. Therefore, the linear relaxations of this model are very weak. Constraints 14 represents in the Wagner's model (1959) as the tardiness of a job *j* is calculated. Where t_{ip} is the start time on machine *i* in position *p*, and α_{ijp} is 1 if job *j* is allocated to position *p* in machine *i*, and 0 otherwise, when the big-*M* constant is activated.

$$T_j \geq t_{ip} + p_{ij} - (1 - \alpha_{ijp})M - d_j, \qquad \forall\, j, p \in N, i \in M \qquad (14)$$

It is proposed to use a dummy variable to occupy empty positions instead of using the big-M constant. Of course, this new variable cannot affect the objective function value. In addition, it is noted that using the position number per machine equal to the job number is a waste of computational effort, since in practice is very unlikely that all jobs will be allocated to same machine. Therefore, an approach has been proposed to reduce the position number per machine. This approach is best explained in Appendix B.

Numerical tests show that the described methodology significantly outperforms the previous models. But for instances with loose due dates, the performance is lower than instances with tight due dates. A method based on Dantzig-Wolfe decomposition was developed to overcome, at least partly, the difficulties associated with the size and loose due date instances, Appendix B. The theoretical basis of the column generation (CG) algorithm has been provided by Dantzig and Wolfe (1960). The linear program is divided into two problems, which are referred to as the relaxed and restricted master problem and subproblem. The subproblem uses the dual information from the master problem to generate new columns that can potentially improve the objective function of the master problem. The iterative process of generating new columns using the subproblem and adding them to the master problem terminates when the subproblem generates new columns with non-negative reduced costs. This approach avoids the difficulty of explicitly generating all columns of the problem. This approach was suggested by Gilmore and Gomory (1961) for solving cutting stock problems.

2.3 Hybridization

In recent years, a lot of attention has been devoted to the integration, or hybridization, of metaheuristics with exact methods. This approach is also called matheuristics which describes works of exploiting mathematical programming techniques in metaheuristic frameworks or on granting to mathematical programming approaches the cross-problem robustness and constrained-CPU-time effectiveness which characterize metaheuristics. Although a relatively new concept, it has been gaining more and more attention over recent years and it has been applied to many combinatorial optimization problems (Fanjul-Peyro et al. 2017). Within scheduling, Billaut et al. (2015) and Fanjul-Peyro et al. (2017) use matheuristic approaches for single machine scheduling and parallel machine scheduling problems with the consideration of additional resources, respectively.

The main motivation behind the hybridization of different algorithms is to exploit the complementary characteristics of different optimization strategies, that is, hybrids are believed to benefit from synergy. In fact, choosing an adequate combination of complementary algorithmic concepts can be the key for achieving top performance in solving many hard optimization problems (Raidl, 2015). Unfortunately, developing an effective hybrid approach is in general a difficult task which requires expertise from different areas of optimization. Moreover, the literature shows that it is nontrivial to generalize, that is, a certain hybrid might work well for specific problems, but it might perform poorly for others. Nevertheless, there are hybridization types that have shown to be successful for many applications. They may serve as guidance for new developments. There are several ways to hybridize metaheuristics with exact methods as explained below (Raidl, 2015).

Intelligent initialization is characterized by supplying a good initial solution. Classic exact optimization methods like branch and bound typically depend on a good initialization in order to be able to prune the search space effectively from the beginning. It is used in both proposed algorithms.

Embedded improvement methods are all strategies that use some method to improve a solution obtained by another method. For example, memetic algorithm which essentially is an evolutionary algorithm including some local improvement technique is applied to all or part the newly created solution candidates of each iteration. Also classical branch and bound often relies on primal solution improvement procedures for finding sooner better primal solutions in order to prune larger parts of search space. It is also used in both algorithms.

Multi-stage approaches are a kind of hybridization to solve problems by decomposing the whole optimization into multiple stages that are addressed by individual techniques in a sequential manner. A first stage of optimization may then be used to fix higher-level decisions while the remaining lower-level variables are determined in one or more successive phases where the higher-level variables are considered fixed. It is used only in second proposed problem.

Large neighborhood exploring approach deals with enhancing a local search using a more efficient algorithm to cover large portions of the search space. Many of combinations of metaheuristics with MILP approaches follow this scheme when a compact MILP model is available for the problem at hand: part of the variables are fixed to the incumbent solution's values and the others are kept open and optimized via a MILP solver. It is also used in the second algorithhm.

Strategic guidance approach is exploit information on promising areas of the search space obtained by other techniques. Problem relaxations are most frequently used for such purposes. If a compact MILP formulation exists for the problem at hand, its linear relaxation often is an obvious choice. An obtained fractional solution can frequently be rounded or in some other way repaired to obtain a feasible integral solution in its proximity. Variables that have already integral values in the relaxed solution might be fixed, or fractional values may be used to bias the search. It is used on second problem.

Given the expected poor performance of Benders decomposition for large problems some heuristics are proposed to accelerate their convergence as done by Lusby et al. (2016). These authors propose an iterative heuristic framework based on Benders decomposition to the shift design problem, which at any given time considers only a very small set of shift types, identified as a good set of shift types.

Techniques described above have inspired the hybridization procedures used in the proposed Benders decomposition algorithm and CG. In the Benders decomposition algorithms is common the initialization with good solutions, called warm start (McDaniel and Devine, 1977). The initialization of the proposed algorithm is based on a restricted version of the original problem that uses linear relaxation information of the problem and a variable number reducing method. The variable number reduction is based on the method called job-machine reduction, proposed by Fanjul-Peyro and Ruiz (2011).

CG has been successfully applied in a number of contexts. But conventional column generation implies slow convergence time. Therefore, column generation has used heuristics and metaheuristics (HCG – Hybrid Column Generation) to accelerate the convergence process. As Beheshti and Hejazi (2015) that use an evolutionary algorithm to solve the subroblems of the vehicle routing problem. Caserta and Voß (2013) use a metaheuristic to solve the subproblems of multi-item multi-period capacitated lot sizing problem with setups. These methods use the PMSP with total tardiness as a criterion.

3. Results

It was written two articles concerned with unrelated parallel machine scheduling problem. The first article: "Improved Combinatorial Benders Decomposition for a Scheduling Problem with Unrelated Parallel Machines" covers the problem for minimization the makespan using mathematical programming and combinatorial Benders decomposition. First, the sub-cycle elimination constraints of a mixed-integer formulation of this problem are removed, creating a problem that is easier to solve. This problem, called the master problem, is decomposed into subproblems that consist of scheduling problems on each machine. The problem can then be solved by a combinatorial Benders decomposition. Thus, an exact approach is generated, where the master problem finds job sequences for machines. However, these job sequences may have sub-cycles. Therefore, for each machine, a subproblem checks the feasibility of the sequence and, if necessary, returns a combinatorial inequality, called the Benders cut, to the master problem. This method can be slow to converge. Therefore, three procedures are introduced to accelerate its convergence. The first procedure is a new method that consists in terminating the execution of the master problem when a repeated optimal solution is found. The second procedure is based on the multi-cut technique and consists of generating several Benders cuts in each iteration based on quality solutions found during the execution of the master problem. The third procedure is based on the warm-start technique and consists in performing a restricted master problem that, because it is easier than the original master problem, is solved in less time, generating Benders cuts more quickly. The improved Benders decomposition scheme is compared to a mathematical formulation and a conventional Benders decomposition algorithm. In the experiments, 600 instances from the literature are tested with up to 60 jobs and 5 machines. The results show that the proposed method failed to find an optimal solution in only 31 instances, obtained an average gap of 0.07%, and took an average of 377.86 s to compute, while the best results of the other methods are 57, 0.17%, and 573.89 s, respectively. This article is presented in Appendix A with some changes in the published version, 3 July 2017 in Journal of Applied Mathematics (https://doi.org/10.1155/2017/9452762).

The second article: "Mathematical Formulation and Hybrid Column Generation for Minimizing Total Tardiness in a Scheduling Problem with Unrelated Parallel Machines" is concerned to the problem considering total tardiness as performance criterion. Mathematical programming and column generation are treated. Mathematical models for this problem often use a big-M due the disjunctive constraints. This yields very weak lower bounds that difficult to obtain an optimal solution even for small size instances. Therefore, it is proposed a new mathematical formulation that does not use the big-M constant. For that, was suggested an approach that uses dummy jobs instead of big-M constant. Experiments conducted on five types of instances show the superiority of the proposed model compared to models based on Wagner's (1959) and Manne's (1960) formulations that use variables based on position and precedence, respectively. Among the computational results the proposed model obtains 291optimal solutions against 98, and 148 of Wagner's (1959) and Manne's (1960) models, respectively, up to three orders of magnitude faster for 300 small size instances tested. In addition, it is proposed a column generation algorithm for finding near-optimal solutions for medium size instances with up to 50 jobs and 10 machines. Unlike standard approaches it is used the proposed model instead of a dynamic programming algorithm to solve the pricing problem. For accelerating the convergence of the column generation algorithm various heuristics are proposed to generate the initial columns and to solve the pricing problem. The hybrid column generation obtains average gap and runtime of 2.71% and 930.48 s, respectively, against 34.78% and 2,490.37 s, respectively, of the proposed model. This article is presented in Appendix B with some changes in the submitted version to Computers and Industrial Engineering Journal, 15 April 2017.

4. General Conclusions and Future Research

The Benders decomposition algorithms (ICBD - proposed method and T&B – Tran and Beck, 2012) presented better performance than the mathematical model from Avalos-Rosales et al. (2015) (MIP), tested in the instances from Vallada and Ruiz (2011) used by Avalos-Rosales et al. (2015). The proposed ICBD method does not find 31 optimal solutions, whereas the T&B and MIP methods, 57 and 63 respectively. The gap and the runtime of the ICBD were 0.07% and 377.86 s, respectively, versus 0.28% and 706.28 s of the MIP and 0.17% and 573.89 s of the T&B, respectively. In addition, the ICBD method in relation to the T&B method also shows improvements compared to the same instances of the paper from Tran and Beck (2012). In this case, as the two methods find optimal solutions for all instances up to the maximum runtime of 3,600 s, the performance difference is in the mean runtime 53.50 s for the ICBD method, versus 65.19 s for T&B method. This indicates that the proposed hybridizations to improve the performance of the Benders decomposition algorithm presented the expected results. Follow others conclusions and proposed future works from the study of this problem.

- The master problem of Benders decomposition algorithms for UPMSP find tight LBs.
- The proposed procedures improve the convergence of the method, especially when the instances are more difficult.
- Using the Benders-and-cut (Geoffrion and Graves, 1974) did not have the expected results.
- The local branching strategy (Fischetti and Lodi, 2003) did not improve the performance of the method.

- Must be develop more efficient Benders cuts than only the no-good cuts.
- Heuristics must be proposed to create good quality solutions to insert in the first iteration of the master problem as a way to reduce the number of solutions that will be evaluated.
- To test the ICBD on similar problems such as vehicle routing.
- Apply ICBD on large parallel machine scheduling problems as a heuristic, restricting the search space to find good quality solutions in less computational time.

The proposed positional model for the UPMSP with total tardiness as the performance criterion presents a tighter linear relaxation than the previous mathematical models that use the big-*M* constant. For example, in the small instances and congestion level q = 5 obtains a gap of 4.40% in relation to the optimal solution, while the Manne model obtains 94.54%. The positional model is easier to find optimal solutions and in less computational time, for example, found optimal solutions for all small instances at the congestion level q = 5 in 2.28 s on average versus 9 optimal solutions found by Manne model in 3,300.28 s with average gap of 63.49%.

The positional model does not perform as well for medium-size instances when the congestion level is low. For example, it obtains a mean gap of 90.22% for congestion level q = 2. Even so, the positional model can be used to solve the subproblems of the proposed Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960). In addition, the proposed column generation algorithm (HCG) used heuristics to generate the initial columns and solve the subproblems. The HCG, for example, for the congestion level q = 1 obtains a mean gap of 0.03% versus 56.77% of the positional model and used 99.30 s of mean runtime versus 2,306.96 s of the positional model. A summary of conclusions and proposed future works from the study of this problem are presented following.

- The positional model that does not need the big-M constant for linearization of the problem precedence constraints has been developed.
- The positional model is able to find good linear relaxation and final solutions compared to the Wagner and Manne models.
- The positional model is not so effective for instances with loose due dates.
- A heuristic based on the linear relaxation of the positional model is possible for tight due dates, that is, q = 3, 4 and 5.
- The positional model efficiently solves the subproblems of the column generation algorithm.
- The iterated local search (ILS) method generates good initial solutions for instances with loose due dates, that is, q = 1 and 2. It allows the hybrid column generation algorithm to be efficient with these instances.

- Can be developed methods to eliminate the negative effect on the performance of the positional model when applied in instances with loose due dates.
- The positional model can be applied in parallel machine scheduling problems with other performance criteria, for example, a combination of completion time and total tardiness.
- To test the reuse of part of the subproblem solutions as well as Desrochers and Soumis (1988) as a way to reduce computational time.
- To test the positional model and HCG on other scheduling problems with the big-M constant in their formulation.
- To apply the idea of dummy job to replace the big-*M* constant in other problems, for example, vehicle routing with pick-up and delivery.

References

ANGHINOLFI, D., & PAOLUCCI, M. (2007). Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Computers & Operations Research*, *34*(11), 3471–3490.

ARMENTANO, V. A., & de FRANÇA FILHO, M. F. (2007). Minimizing total tardiness in parallel machines scheduling with setup times: an adaptive memory-based GRASP approach. *European Journal of Operational Research*, *183*(1), 100–114.

ARNAOUT, J.-P., MUSA, R., & RABADI, G. (2014). A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines—part II: Enhancements and experimentations. *Journal of Intelligent Manufacturing*, 25(1), 43–53.

AVALOS-ROSALES, O., ANGEL-BELLO, F., & ALVARES, A. M. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *The International Journal of Advanced Manufacturing Technology*. 76(9), 1705-1718.

BEHESHTI, A. K., & HEJAZI, S. R (2015). A novel hybrid column generation-metaheuristic approach for the vehicle routing problem with general soft time window. *Information Sciences*, 316, 598–615.

BILGE, U., KIRAC, F., KURTULAN, M., & PEKGUN, P. (2004). A tabu search algorithm for parallel machine total tardiness problem. *Computers & Operations Research, 31*(3), 397–414.

BILLAUT, J. C., CROCE, F. D., & GROSSO, A. (2015). A single machine scheduling prob-lem with two-dimensional vector packing constraints. *European Journal of Operational Research*, 243(1), 75–81.

CAMM, J. D., RATURI, A. S., & TSUBAKITANI, S. (1990). Cutting big M down to size. *Interfaces*, 20(5), 61-66.

CASERTA, M., & VOß, S. (2013). A math-heuristic Dantzig-Wolfe algorithm for capacitated lot sizing. *Annals of Mathematics and Artificial Intelligence,* 69(2), 207–224.

CHEN, C.-L. (2012). Iterated hybrid metaheuristic algorithms for unrelated parallel machines problem with unequal ready times and sequence-dependent setup times. *The International Journal of Advanced Manufacturing Tecnology*, *60*(5), 693–705.

CHEN, J. F. (2009). Scheduling on unrelated parallel machines with sequence- and machine-dependent setup times and due-date constraints. *The International Journal of Advanced Manufacturing Tecnology*, *44*(11), 1204–1212.

CHEN, C. L., & CHEN, C. L. (2009). Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, *43*(1-2), 161–169.

CHEN, Z. L., & POWELL, W. B. (1999). Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing*, *11*(1), 78–94.

DANTIZIG, G. B., & WOLFE, P. (1960). Decomposition principle for linear programs. *Operations Research*, *8*(1), 101–111.

DESROCHERS, M., & SOUMIS, F. (1988). A reoptimization algorithm for the shortest path problem with time windows. *European Journal of Operational Research*, *35*(2), 242–254.

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(2-3), 255-270.

FANJUL-PEYRO, L., PEREA, F., & RUIZ, R. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *European Journal of Operational Research*, 260(2), 482–493.

FANJUL-PEYRO, L., & RUIZ, R. (2011). Size-reduction heuristics for the unrelated parallel machines scheduling problem, *Computers & Operations Research*, 38(1), 301–309.

FISCHETTI, M., & LODI, A. (2003). Local branching. *Mathematical Programming*, Series B, 98, 23–47.

FLESZAR, K., CHARALAMBOUS, C., & HINDI, K. S. (2012). A variable neighborhood descent heuristic for the problem of makespan minimization on unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*, 23(5), 1949–1958.

GEOFFRION, A. M., GRAVES, G. W. (1974). Multicommodity distribution system design by benders decomposition. *Management Science*, 20(5), 822–844.

GILMORE, P. C., & GOMORY, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6), 849–859.

GRAHAM, R. L., LAWWLER, E. L., LENSTRA, J. K., & KAN, A. H. G. R. (1979) Optimization and approximation in deterministic sequencing and scheduling: A survey *Annals of Discrete Mathematics*, 5(2), 287–326.

GUPTA, J. N. D., & STAFFORD Jr., E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3), 699–711.

KLOTZ, E., & NEWMANN, A. M. (2013). Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1-2), 18-32.

LEE, S., & GROSSMANN, I. E. (2000). New algorithms for nonlinear generalized disjunctive programming. *Computers and Chemical Engineering*, 24(9-10), 2125-2141.

LEE, J. H., YU, J. M., & LEE, D. H. (2013). A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness. *The International Journal of Advanced Manufacturing Technology*, *69*(9), 2081–2089.

LIN, S. W, LU, C. C., & YING, K. C. (2011). Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints. *The International Journal of Advanced Manufacturing Technology*, *53*(1), 353–361.

LOPES, M. J. P., & de CARVALHO, J. M. V. (2007). A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. *European Journal of Operational Research*, *176*(3), 1508-1527.

LUSBY, R. M., RANGE, T. M., & LARSEN, J. (2016). A Benders decomposition-based matheuristic for the Cardinality Constrained Shift Design Problem. *European Journal of Operational Research*, 254(2), 385–397.

McDANIEL, D., & DEVINE, M. (1977). A modified Benders partitioning algorithm for mixed integer programming. Management Science, 24(3), 312–319.

M'HALLAH, R., & BULFIN, R. L. (2005). Minimizing the weighted number of tardy jobs on parallel processors. *European Journal of Operational Research*, *160*(2), 471–484.

MOKOTOFF, E. (2001). Parallel machine scheduling problems: a survey. *Asia-Pacific Journal of Operational Research*, 18(2), 193–242.

NOGUEIRA, J. P. de C. M., ARROYO, J. E. C., VILLADIEGO, H. M. M., & GONÇALVES, L. B. (2014). Hybrid GRASP Heuristics to Solve an Unrelated Parallel Machine Scheduling Problem with Earliness and Tardiness Penalties. *Electronic Notes in Theoretical Computer Science*, *302*, 53–72.

PAULA, M. R., MATEUS, G. R., & RAVETTI, M. G. (2010). A non-delayed relax-and-cut algorithm for scheduling problems with parallel machines, due dates and sequence-dependent setup times. *Computers & Operations Research*, *37*(5), 938–949.

PAULA, M. R., RAVETTI, M. G., MATEUS, G. R., & PARDALOS, P. M. (2007). Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search. *IMA Journal of Management Mathematics*, 18(2), 101–115.

PINEDO, M. L. (2012). Scheduling: theory, algorithm, and systems. New York: Springer-Verlag, 4 ed., 676 pp.

RAIDL, G. R. (2015). Decomposition based hybrid metaheuristics. *European Journal of Operational Research*, 244(1), 66–76.

ROCHA, P. L., RAVETTI, M. G., MATEUS, G. R., & PARDALOS, P. M. (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers & Operations Research*, *35*(4), 1250–1264.

TRAN, T., & BECK, J. C. (2012). Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups. In Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012), 774–779.

VALLADA, E., & RUIZ, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, vol. 211, no. 3, pp. 612–622.

VAN DEN AKKER, J. M., HURKENS, C. A. J., & SAVELSBERGH, M. W. P. (2000). Time-Indexed Formulations for Machine Scheduling Problems: Column Generation. *INFORMS Journal on Computing*, 12(2), 111–124.

WAGNER, H. W. (1959). An integer linear-programming model for machine scheduling. *Naval Research Logistic Quarterly*, *6*(2), 131–140.

ZEIDI, J. R., & HOSSEINI, S. M. (2015). Scheduling unrelated parallel machines with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology, 81*(9), 1487–1496.

YING, K., LEE, Z., & LIN, S. (2012). Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*, 23(5), 1795–1803.

Appendix A

Improved Combinatorial Benders Decomposition for a Scheduling Problem with Unrelated Parallel Machines

Francisco Regis Abreu Gomes

Graduate Program in Production Engineering, Federal University of Minas Gerais, Brazil and Federal Institute of Education, Science and Technology of Ceará, Brazil

Geraldo Robson Mateus

Computer Science Department, Federal University of Minas Gerais, Brazil

Abstract

This paper addresses the unrelated parallel machines scheduling problem with sequence and machine dependent setup times. Its goal is to minimize the makespan. The problem is solved by a combinatorial Benders decomposition. This method can be slow to converge. Therefore, three procedures are introduced to accelerate its convergence. The first procedure is a new method that consists of terminating the execution of the master problem when a repeated optimal solution is found. The second procedure is based on the multicut technique. The third procedure is based on the warm-start. The improved Benders decomposition scheme is compared to a mathematical formulation and a standard implementation of Benders decomposition algorithm. In the experiments, two test sets from the literature are used, with 240 and 600 instances with up to 60 jobs and 5 machines. For the first set the proposed method performs 21.85% on average faster than the standard implementation of the Benders algorithm. For the second set the proposed method failed to find an optimal solution in only 31 in 600 instances, obtained an average gap of 0.07%, and took an average computational time of 377.86 s, while the best results of the other methods were 57, 0.17%, and 573.89 s, respectively.

Keywords: scheduling; unrelated parallel machines; combinatorial Benders decomposition; acceleration techniques.

1. Introduction

This paper addresses the unrelated parallel machines scheduling problem with sequence and machine dependent setup times (UPMSP-SMDST). Scheduling problems with parallel machines have been extensively studied and applied in many manufacturing systems [1]. Because of the rising costs of raw materials, labor, energy, and transportation, the role of scheduling is currently essential for the planning of companies [2]. To learn more about these kinds of problems, the survey produced by Mokotoff [3] can be consulted. Most of the literature on these problems ignores the setup time between jobs. However, Allahverdi and Soroush [4] presented a study that shows the importance of considering the setup time to produce more realistic and effective planning.

The UPMSP-SMDST is an NP-hard problem, since a special case of this problem with a single machine is equivalent to the traveling salesman problem, which is NP-hard [5]. Among the few studies that use exact methods for the solution of this problem, Rocha et al. [6] is notable because it proposed a branch-and-bound approach to minimize the makespan and the sum of weighted tardiness of each job. Paula et al. [7] proposed a non-delayed relax-and-cut algorithm based on the Lagrangian relaxation of a time-indexed formulation to minimize the total weighted tardiness. Tran and Beck [8] presented an algorithm based on a logic-based Benders decomposition to minimize the makespan. Finally, Avalos-Rosales et al. [1] explored many mathematical formulations with a new linearization to calculate the makespan.

Most studies use heuristics and metaheuristics to solve the UPMSP-SMDST. Among these papers, Paula et al. [9] proposed an approach based on variable neighborhood search to minimize the makespan and the sum of weighted tardiness of each job. Lin et al. [10] presented an iterated greedy heuristic to minimize the total tardiness. Vallada and Ruiz [11] used two versions of a genetic algorithm to minimize the makespan. Ying et al. [12] proposed a restricted simulated annealing algorithm to minimize the makespan, and Lee et al. [13] evaluated an algorithm based on the tabu search to minimize the total tardiness. Arnout et al. [14] presented an ant colony algorithm with two stages to minimize the makespan. Finally, Avalos-Rosales et al. [1] proposed three versions of a method based on multi-start and VNDS algorithms to minimize the makespan.

The combinatorial Benders decomposition was chosen to solve the UPMSP-SMDST in this study because it has been successfully applied to several scheduling problems ([15], [16], [17], [18], [8], [19]. The Benders decomposition method consists in dividing the original problem into a master problem and an easier subproblem. In a minimization problem, the master problem solution provides a lower bound (LB) and the subproblem solution provides an upper bound (UB) to the original problem. The subproblem is used to evaluate the feasibility of the solutions provided by the master problem and, if necessary, generate combinatorial inequalities, called Benders cuts, which are added to the master problem iteratively until the optimal solution of original problem is obtained [20]. As the Benders cuts are added, the difference between the UB and LB decreases, and when $UB - LB \leq \varepsilon$, where ε is some tolerance, the optimal solution has been found. This method is also known as the logic-based Benders decomposition. The combinatorial Benders decomposition is a generalization of the classic Benders decomposition because the subproblem may be any combinatorial problem, not necessarily a linear or nonlinear programming problem [21].

The contribution of this paper is the proposal of three procedures to accelerate the convergence of the combinatorial Benders decomposition as applied to the UPMSP-SMDST. The first procedure is proposed for the first time and consists of terminating the execution of the master problem when a repeated optimal solution is found. The second procedure is based on the multi-cut technique and generates several Benders cuts at each iteration based on quality solutions found during the execution of the master problem. The third procedure is based on the warm-start technique and consists of performing a restricted master problem that is easier and hence quicker to solve than the original master problem, generating Benders cuts more quickly. Moreover, with specific adaptations, these procedures may be applied to other problems.

The rest of the paper is organized as follows. Section 2 presents the definition and an actual mathematical formulation of the UPMSP-SMDST. Section 3 presents a definition of the Benders decomposition and reviews papers on convergence acceleration techniques for this method. It also describes the proposed procedures and their combination to create the method proposed in this paper, which is called the improved combinatorial Benders decomposition (ICBD). Section 4 presents the results of computational experiments, which compare the best reported mathematical formulation, standard implementation of Benders decomposition, and ICBD. In Section 5, the conclusions are presented.

2. Problem formulation

In UPMSP-SMDST, a set N of jobs is scheduled on a set M of machines. Each job j takes processing time p_{ij} on machine i. The system machines are unrelated, which means that job j can have a processing time that is longer than job k on a specific machine, although the same cannot be true for another machine. There is a setup time, s_{ijk}, which corresponds to the time required between the end of job j and the beginning of job k on machine i. In this model, it is necessary to use a dummy job 0, with all its parameters equal to zero. Moreover, it is the first and last jobs of the sequences, where N_0 is the set of jobs plus dummy job 0. The goal of the problem is to determine a schedule of job assignments for the machines that minimizes the makespan. Using the three-element notation of Graham et al. [22], this problem can be classified as $R \mid sds \mid C_{max}$.

Currently, the best mathematical model for the UPMSP-SMDST was proposed by Avalos-Rosales et al. [1]. In this model, y_{ij} is 1 if job j is processed in machine i (and 0, otherwise), x_{ijk} is 1 if the job k is processed immediately after job j in machine i (and 0, otherwise), C_j denotes the completion time of job j, and C_{max} is the makespan of the solution. The model itself is as follows:

$$\min C_{max} \tag{1}$$

$Subject\ to$:

$$\sum_{i \in M} y_{ij} = 1, \qquad \forall j \in N \tag{2}$$

$$\sum_{\substack{j \in N_0 \\ j \neq k}} x_{ijk} = y_{ik}, \qquad \forall k \in N, i \in M \tag{3}$$

$$\sum_{\substack{k \in N_0 \\ j \neq k}} x_{ijk} = y_{ij}, \qquad \forall j \in N, i \in M \tag{4}$$

$$\sum_{k \in N} x_{i0k} \leq 1, \qquad \forall i \in M \qquad (5)$$

$$C_{max} \geq \sum_{\substack{j \in N_0 \\ j \neq k}} \sum_{k \in N} s_{ijk} x_{ijk} + \sum_{j \in N} p_{ij} y_{ij}, \qquad \forall i \in M \qquad (6)$$

$$C_k \geq C_j + s_{ijk} + p_{ik} - (1 - x_{ijk})V, \qquad \forall j \in N_0, k \in N, j \neq k, i \in M \qquad (7)$$

$$C_{max} \geq C_j, \qquad \forall j \in N \qquad (8)$$

$$C_0 = 0, \qquad (9)$$

$$C_j \geq 0, \qquad \forall j \in N \qquad (10)$$

$$x_{ijk} \in \{0,1\}, \qquad \forall j, k \in N_0, j \neq k, i \in M \qquad (11)$$

$$y_{ij} \in \{0,1\}, \qquad \forall j \in N, i \in M \qquad (12)$$

Objective (1) minimizes the makespan of the solution. Constraints (2) ensure that each job is processed by only one machine. Constraints (3) and (4) ensure that each job has only one predecessor and successor, respectively. Constraints (5) ensure that a maximum of one job is scheduled as the first job on each machine. Constraints (6) are a new linearization to calculate makespan that is independent of *V*, which is a very high value constant. Although, it is worth noting that Tran and Beck [8] were the first to propose this constraint to strengthen the master problem. Constraints (7) ensure the correct order of the jobs and eliminate the formation of sub-cycles, that is, if $x_{ijk} = 1$ the completion time of job *k* should be greater or equal than the completion time of job *j*, and if $x_{ijk} = 0$, these constraints becomes redundant. Constraints (8) also define the makespan of the solution. Constraints (9) assign 0 to the completion time of the dummy job. Constraints (10) to (12) define the non-negativity and integrality of the variables. Finally, constraints (6) are responsible for the efficiency of this model relative to other models applied to this problem.

3. Combinatorial Benders Decomposition

The combinatorial Benders decomposition can be used to decompose the UPMSP-SMDST into a master problem of job allocation on machines and *m* scheduling subproblems on a single machine. The subproblems are used to evaluate the feasibility of the solutions found by the master problem and to generate Benders cuts if needed. A standard implementation of this method for the UPMSP-SMDST was first proposed by Tran and Beck [8]. However, the direct application of this method converges slowly. Therefore, this paper proposes three procedures for accelerating its convergence.

The main issues associated with this slow convergence are (i) the run times of the master problem and subproblems, and (ii) the quality of the produced cuts [23]. Many studies have been carried out to develop techniques to accelerate the convergence of the Benders decomposition. They can be classified into two main approaches. The first uses strategies to reduce the computational effort to solve the master problem, and the second generates more effective cuts to eliminate infeasible or suboptimal solutions [24]. Because the Benders cuts generated from any solution of the master problem are valid, this enables the creation of many types of cuts ([25], [26], [27], [23], [24], [28], [29]). McDaniel and Devine [30] suggested the warm-start technique, which generates cuts using the solution of the master problem by relaxing the integer variables. Wheatley et al. [31] developed a scheme called restrict-and-decompose, which consists of relaxing the integer variables of the original master problem and executing it. When this problem does not generate more cuts, the technique returns to the original master problem. Geoffrion and Graves [32] proposed a scheme in which Benders cuts are generated each time a new feasible solution that is better than the current incumbent solution is found. This strategy avoids having to solve the master problem until the end in order to generate Benders cuts. It can also economize computational time. Cote and Laughton [33] demonstrated the benefit of using a heuristic to find good solutions to the master problem. Similarly, Rei et al. [26] used the local branching strategy of Fischetti and Lodi [34] to explore the neighborhood of each solution obtained by the master problem to detect repeated optimal solutions. Poojari and Beasley [35] used a genetic algorithm along with a heuristic to find feasible solutions of the master problem. Sherali and Lunday [24] proposed generating a set of initial cuts for the master problem. Huang and Zheng [36] proposed a type of feasibility cut to iteratively remove infeasible solutions with certain characteristics. Another strategy is to propose and introduce valid inequalities in the master problem before starting the method in order to eliminate infeasible solutions ([27], [37], [38]). Generating more than one good quality Benders cut in each iteration is known as the multi-cut technique [39]. Magnanti and Wong [40] defined the concept of Pareto-optimal cut for degenerate Benders subproblems and applied the multi-cut technique.

The proposed acceleration procedures are aimed at reducing the execution time of the master problem and multi-cut generation. Methods to improve the quality and quantity of cuts generated as Pareto-optimal cut [40], covering cut bundle [27], maximum feasible subsystem [23], maximization density cut [28], among other methods cited were not used because they depend of a linear subproblem, and in the problem under study the subproblem is integer. The valid inequalities found in the literature were also not used because they are specific to the problems addressed, from an depth analysis we did not identify any specific or generic valid inequality for the studied problem. Furthermore, we tested two acceleration methods cited, the first from Geoffrion and Graves [32], and second the local branching strategy from Fischetti and Lodi [34]. But, they failed to have a better performance than the procedures that we propose. The proposed convergence acceleration procedures are as follows.

3.1 Termination of the master problem execution

In a standard Benders decomposition sometimes the optimal solution of the master problem (LB) is equal to the optimal solution of the previous iteration, that is, different solutions with the same value. Therefore, we propose a procedure that terminates the execution of the master problem early when a repeated optimal solution is found. Hence, when this happens, the master problem does not need to run to the end, saving computational time.

Proposition 1. *If, during the master problem execution, a new solution equal to the current LB is found, the execution of the master problem is terminated, and the LB keeps the same value.*

Proof. The optimal solution value of the master problem cannot be lower than the value of the optimal solution found in the previous iteration. That is, given the lower bound at iteration k (LB_K), by definition the sequence of lower bounds obtained by the master problem is $LB_1 \leq LB_2 \leq LB_3... \leq LB_k$. Otherwise, the previous solution would not be optimal. This occurs because there can be multiple optimal solutions with the same value. Therefore, in this case, the LB remains the same.

3.2 Multi-cuts

A combinatorial Benders cut (CBC) is generated when an infeasible subproblem is identified. In the problem in this study, this happens when the master problem finds a job sequence for machine i with sub-cycles. For instance, consider six jobs labeled 1 to 6. Two sub-cycles would be 0-1-3-4-0 and 6-5-2-6. Tran and Beck [8] proposed the cut shown below.

$$LB \geq C_i^h - \sum_{j \in N_i^h} (1 - y_{ij})\, \theta_{hij,} \tag{13}$$

where C_i^h is the completion time of the jobs in the subproblem associated with machine i at iteration h, N_i^h is the set of jobs assigned to machine i at iteration h, and θ_{hij} is an upper bound of the effect of job j on completion time when assigned to machine i at iteration h, calculated as $p_{ij} + max\,(s_{ikj})$, $k \in N_i^h, k \neq j$. That is, when job j is no longer part of the solution, the value of LB can be reduced up to θ_{hij}.

By analyzing the proposed cut by Tran and Back [8] there is a failure. Given the hypothetical job sequence S={a, b, c, d}, if the job c was removed, the effect on LB only by the setup times is s_{ibc} + s_{icd} – s_{ibd}, and if max(s_{ikc}) ≥ s_{ibc} + s_{icd} – s_{ibd}, the cut is still valid, otherwise it is not. Therefore, we use a "no-good" cut that only eliminates an infeasible solution that has been found. According to some authors, this type of cut can be very weak [41], but it was used because a no special structure was found that could build stronger cuts. That is, cuts that eliminate other infeasible or suboptimal solutions. The only change made in relation to (13) was to replace θ_{hij} by a very high value constant. Tests with the version of Tran and Beck' Benders algorithm using both cut types showed no difference in performance and solutions obtained. We made this change in cut because the previous cut is not a separation cut as claimed, but only a no-good cut.

Experiments carried out with the standard implementation of Benders decomposition have shown that the master problem generated many quality solutions in addition to the optimal solution. Given an iteration h, we define quality solutions those that have a value S, $LB \leq S \leq UB^{h-1}$. The optimal solution of the next iteration (h + 1) may be among these solutions, if the method has not been terminated in iteration h. Therefore, these solutions, including the optimal solution, are stored in a set called as solutions pool. For example, if the values of LB = 120 and UBh-1 = 150, and the solutions returned by the solver after solving the master problem are {s0 = 122, s1 = 131, s2 = 141, s3 = 147, s4 = 152}. The pool will be formed by the solutions s0 to s3, because they are equal and smaller to UBh-1. Each solution of the pool is solved by the subproblem, not just the optimal solution, which is why the procedure is a multi-cut. When a job sequence of a machine in the solution pool is found to be infeasible, a CBC is generated, as described above. This forces the master problem to generate solutions other than those of the solutions pool in the next iteration. Thus, the multi-cut strategy reduces the number of iterations required for the convergence of the method, thereby reducing computational runtime.

3.3 Warm-start

A warm-start procedure for the combinatorial Benders decomposition is proposed, based on the idea of solving a restricted master problem. The aim is to produce good quality CBCs more quickly. Many authors have shown that the strong lower bounds found by the linear relaxations of time-indexed formulations for machine scheduling problems provide useful information for guiding primal heuristics called list-scheduling algorithms ([42], [43], [44]). In this sense, the linear relaxation of the Benders decomposition master problem also provides a strong lower bound. The tests conducted in this study show that the gap between the linear relaxation and integer optimal solution of the master problem was on average 7%. In addition, Fanjul-Peyro and Ruiz [45] showed that, for a scheduling problem on parallel machines without setup time, size-reduction heuristics produce good quality solutions with little computational effort. These heuristics use some clever criteria to reduce the number of variables available during the run of the mathematical model. We join these two ideas to propose our restricted master problem.

The restricted master problem is obtained by setting a set of variables of the master problem to zero as follows. First, a linear relaxation of the master problem is performed. That is, all jobs for which the variable y_{ij} obtained a nonzero value are inserted into the set of jobs available for machine i, which is denoted as N_i^R. In addition, the rest of the jobs are inserted into the set of jobs not available for machine i, denoted by N_i^O. Thus, the restricted master problem is executed with the variables y_{ij} of the jobs in N_i^O set to 0, that is, they cannot be chosen, while the variables y_{ij} of the jobs in N_i^R can take the value of 0 or 1. To increase the number of available jobs on each machine and consequently improve the quality of the solutions, the following size-reduction heuristic is used. We first evaluate each job in N_i^O and choose the one that could possibly generate the least effect on the completion time of machine i (C_i), which is then inserted into N_i^R. To calculate this effect, parameter δ_{ik} is calculated for each job $k \in N_i^O$. This parameter is the sum of the processing time of job k on machine i, the lowest setup time for jobs j subsequent to job k, and the lowest setup time for jobs j before job k, where $j \in N_i^R$, that is, $\delta_{ik} = p_{ik} + \min(s_{ikj}) + \min(s_{ijk})$. The job k with the minimum δ_{ik} is inserted into N_i^R and removed from N_i^O. This procedure is repeated until N_i^R achieves the desired size.

The proposed warm-start procedure consists of a Benders decomposition using the restricted master problem described above rather than the master problem with all available jobs (the original master problem). The master problem is hence solved more quickly, and thus CBC are also generated more quickly. The warm-start procedure is executed in two stages with different percentages of jobs in N_i^R, because they empirically showed better performance. Each stage ends after a fixed number of iterations or when the optimal solution of the restricted master problem is equal to the UB. We make the observation that the optimal solution of the restricted master problem is not an LB of the original problem.

3.4 ICBD

The master problem is a relaxation of the mixed-integer formulation proposed by Avalos-Rosales et al. [1] for the UPMSP-SMDST. This relaxation removes the elimination constraints of the sub-cycles, i.e., constraints (7), and consequently constraints (8) and (10). For this reason, the master problem may find job sequences with sub-cycles, which are infeasible solutions. However, this relaxation provides a tight LB and is significantly easier to solve than the complete problem. Thus, this relaxation decomposes the UPMSP-SMDST into a master problem of job allocations and *m* scheduling subproblems on a single machine, which are used to evaluate the existence of sub-cycles.

Given a solution of the master problem, where C_i^{mp} is the completion time of the job sequence of machine *i* in the master problem, the next step is to determine the existence of any sub-cycles on each machine *i* by means of a subproblem. The resulting subproblem is equivalent to the traveling salesman problem with directed arcs, also known as asymmetric traveling salesman problem. In this representation, the jobs are the nodes and the distances between the nodes are the setup times between jobs. The completion time of the sequence is the sum of the distances between the nodes and the processing time of the jobs. For each iteration h of the algorithm and machine *i*, one subproblem SP_i^h is generated and its completion time C_i^h is found. When $C_i^h > C_i^{mp}$, the sequence has a sub-cycle, so a CBC is generated and added to the master problem. The biggest C_i^h is the iteration makespan C_{max}^h. If C_{max}^h is smaller than the UB, then it becomes the new UB. This procedure is called subproblem evaluation, and its pseudocode is shown in Algorithm 1.

Algorithm 1: Subproblem evaluation.

```
1  Given a solution of the master problem;
2  $C_{max}^h \leftarrow 0$;
3  **for** *i*=1 until *m* **do**
4    $C_i^h \leftarrow$solve $SP_i^h$;
5    **if** $C_i^h > C_i^{mp}$ (has sub-cycle) **then** add CBC;
6    **if** $C_i^h > C_{max}^h$ **then** $C_{max}^h \leftarrow C_i^h$;
7  **end-for**
8  **if** UB>$C_{max}^h$ **then** UB $\leftarrow C_{max}^h$;
```

The proposed ICBD method consists of solving the master problem (MP) using the three proposed procedures and the subproblems until a terminating condition is true. In each iteration h of ICBD, the master problem generates a solution pool of size |Pool| according to the multi-cut procedure outlined in Section 3.2. Algorithm 1 evaluates each one of the solutions. The ICBD algorithm is presented in Algorithm 2.

Algorithm 2: ICBD

```
1  begin
2    $h \leftarrow 0$; UB ← +∞; stop ← false;
3    while (stop = false) do
4      $h \leftarrow h + 1$;
5      solve MP*;
6      for $k$=1 until |Pool| do // multi-cut
7        evaluation of subproblems (Algorithm 1);
8      end-for
9      evaluate the terminating condition;
10   end-while
11 end
```

* Restricted or original master problem.

Algorithm 2 is used for both the restricted and original master problems. Thus, this algorithm is executed twice in sequence: once in the warm-start procedure with the restricted master problem, and once with the original master problem. The warm-start procedure is terminated at the conclusion of its two stages or when their execution time reaches the maximum time allowed. The original master problem is terminated when the optimality condition (UB – LB ≤ 0.0001) or total allowed run time is reached. It is important to note that during the warm-start procedure, the optimal solution of the master problem is not a valid LB of the problem because it does not have all the variables available.

4. Computational experiments

In order to test the mathematical formulation and Benders decomposition methods, they were implemented using API Concert Technology for C++ and solved using IBM ILOG CPLEX 12.5. Tests were performed on a Dell Inspiron notebook, equipped with an Intel Core i5-2430M 2.40 GHz processor with 4 GB of memory and a Windows 7 operating system. The maximum runtime allowed for any case was 3,600 s. If the solver was not able to find the optimal solution, the best integer solution obtained is reported.

The computational experiments are performed using two different instance sets. First with the instances used by Tran and Beck [8] and next with instances from Vallada and Ruiz [11] used by Avalos-Rosales et al. [1]. The test instances obtained from Tran and Beck [8] have the following configuration, with number of jobs $N \in \{10, 20, 30, 40, 50, 60\}$ and number of machines $M \in \{2, 3, 4, 5\}$. Setup times were uniformly distributed at the interval: 25–50. Processing times were uniformly distributed between 5 and 200. There were 10 replications for each possible combination of numbers of job and machine, making a total of 240 instances. According to Tran and Beck (2012) to obtain setup times that were sequence dependent and follow the triangular inequality, each job was given two different sets of coordinates on a Cartesian plane for every machine. The setup times are the Manhattan distances between two jobs' coordinates. Distances between the second set of coordinates are used to provide asymmetric setup times. The triangular inequality states that, for any three jobs j, l, k requiring the same resource (machine i), the inequality $s_{ijk} \leq s_{ijl} + p_{il} + s_{ilk}$. The test instances obtained from Vallada and Ruiz (2011), $N \in \{20, 30, 40, 50, 60\}$ and $M \in \{2, 3, 4, 5\}$. Setup times were uniformly distributed over three intervals: 1–49, 1–99, and 1–124. Processing times were uniformly distributed between 1 and 99. There were 10 replications for each possible combination of jobs and machines, and setup time, making a total of 600 instances. These latter authors did not state whether the triangular inequality of their instances is ensured. Last instances are available at http://soa.iti.es.

The instances were grouped by number of jobs and machines. Therefore, each table row represents the average results of 10 or 30 instances tested from Beck and Tran [8] or Vallada and Ruiz [11], respectively. Table 1 compares the results of the Benders decomposition method of Tran and Beck [8] (T&B), and the proposed ICBD method using instances from Tran and Beck [8]. Columns 1 and 2 refer to the number of jobs and machines, respectively. The remainder of the table is divided into three groups. The first group refers to the average percentage gap between the first iteration LB of MP (LB_1) and the optimal solution (opt), wich is calculed as $100*(opt – LB_1)/LB_1$. The second and third group show the results from T&B and ICBD methods. Columns of each group refer to the number of iterations (iter), number of cuts (#cut) and run time (time).

All instances from Tran and Beck [8] were solved to optimality by the two methods in less than 3,600 s. From Table 1 it is noted that the average number of iterations of T&B method was 1.69. A more detailed analysis showed that 44.2% of instances are solved with only one iteration (i.e., the first solution of MP is equal to optimal solution) and 45.8% of instances are solved in two iterations. Although, 90% of instances are solved within two iterations, and the maximum number of iterations was 5 wich occurred once. Therefore, with this instance set the ICBD method was performed using only the multi-cut procedure because other procedures only consume computational time and not bring any advantage. In the combinations 10x2, 10x3, 10x4, and 20x4 the ICBD method does not reduce the number of iterations or increase the number of cuts generated. In other combinations there were improvements, but as the number of iterations is small the improvements are also small. The biggest differences in runtimes were in the instances with five machines, usually more difficult. Moreover, the final reduction in runtime using the ICBD method compared to T&B method was 21.85%.

Table 1 – Comparison of T&B, and ICBD methods using the instances from Tran and Beck (2012).

n	m	LB		T&B			ICBD		
		% gap	time	# iter	# cut	Time	# iter	# cut	Time
10	2	0.23	0.05	1.40	2.80	0.10	1.40	2.80	0.11
	3	0.14	0.11	1.20	3.60	0.16	1.20	3.60	0.17
	4	0.44	0.23	1.10	4.40	0.30	1.10	4.40	0.29
	5	0.40	0.23	1.20	6.00	0.34	1.20	6.50	0.33
20	2	0.11	0.18	1.80	3.60	0.45	1.70	4.00	0.42
	3	0.34	0.40	2.10	6.30	0.90	2.00	8.70	0.86
	4	0.18	1.02	1.50	6.00	1.55	1.50	6.00	1.58
	5	0.31	1.85	1.50	7.50	2.79	1.40	7.00	2.59
30	2	0.07	0.33	1.70	3.40	0.71	1.60	3.40	0.66
	3	0.17	0.94	1.80	5.40	1.70	1.70	6.60	1.63
	4	0.27	2.56	1.90	7.60	7.01	1.90	10.80	6.87
	5	0.17	18.19	1.80	9.00	99.49	1.70	10.00	68.15
40	2	0.06	0.72	1.90	3.80	2.02	1.90	4.00	1.94
	3	0.05	1.57	1.60	4.80	2.77	1.50	5.40	2.73
	4	0.17	11.73	1.80	7.20	18.77	1.70	8.00	18.66
	5	0.19	41.14	1.90	9.50	102.10	1.70	12.00	99.24
50	2	0.03	0.69	1.80	3.60	2.23	1.70	4.00	2.14
	3	0.07	3.37	1.50	4.50	6.42	1.50	4.80	5.24
	4	0.12	28.65	1.70	6.80	44.32	1.70	9.20	40.59
	5	0.09	133.74	1.70	8.50	267.45	1.70	12.00	212.99
60	2	0.04	1.44	1.80	3.60	3.89	1.70	3.80	3.79
	3	0.05	5.66	1.80	5.40	8.89	1.70	5.70	8.58
	4	0.05	64.36	1.70	6.80	111.89	1.60	8.00	106.71
	5	0.13	303.25	2.30	11.50	878.30	2.10	15.00	697.76
Average		0.16	25.93	1.69	5.90	65.19	1.62	6.90	53.50

Table 2 shows the average percentage gap of results obtained using the first solution of MP and the optimal solution (or the best solution) for the instances from Vallada and Ruiz [11]. The first column represents the number of jobs, remainder of the table is divided into five groups, the first four groups show the results for four number of machines and average results for each number of jobs is shown in the fifth group. It is noted that the gaps are greater than those obtained using the instances from Tran and Beck [8], an overall average gap of 1.54% vs. 0.16%, respectively. Gaps increase as the number of machines also increases, but the opposite occurs when increasing the number of jobs.

Table 2 – Results of average percentage gap for the first MP solution using instances from Vallada and Ruiz (2011).

n	m = 2		m = 3		m = 4		m = 5		Average	
	%gap	time	%gap	time	%gap	time	%gap	time	%gap	time
20	1.22	0.17	2.56	0.39	3.00	1.04	3.15	1.70	2.48	0.82
30	0.88	0.33	1.56	0.96	1.88	3.56	2.51	10.50	1.71	3.84
40	0.56	0.58	1.21	1.41	1.70	7.07	1.98	36.38	1.36	11.36
50	0.34	1.19	0.76	2.30	1.30	11.92	2.16	106.81	1.14	30.56
60	0.32	1.23	0.70	3.47	1.12	31.44	1.87	228.16	1.00	66.08
Average	0.66	0.70	1.36	1.71	1.80	11.01	2.33	76.71		

The parameter values used by ICBD method are shown next. The percentages of jobs in the sets N_i^R in the warm-start procedure were set to 50% and 75%, for the first and second stages, respectively. The maximum number of iterations of each warm-start stage was eight. A calibration of these parameters was attempted, although in the combinations tested, none had a superior statistical performance, so these tests are not presented. The maximum time allowed for the execution of the two warm-start stages was 1,800 s. The maximum execution time of the original master problem was 3,600 s minus the total execution time of the warm-start procedure.

Table 3 compares the results of the mixed-integer programming model (MIP) of Avalos-Rosales et al. [1], the T&B method, and the ICBD (with three proposed procedures) method using the instances from Vallada and Ruiz [11]. Columns 1 and 2 refer to the number of jobs and machines, respectively. The remainder of the table is divided into three groups. The first group refers to the number of unsolved instances until optimality (#Uns.). The second group shows the average percentage gap (%Gap), which is calculated as 100*(UB – LB)/LB. The third group shows the average CPU time elapsed in seconds (Time) when solving the instances. There are three columns for each group and one for each method evaluated. Values in italics indicate the best result for a particular combination of jobs and machines.

Comparing the three methods, ICBD obtained the best results for each one of the three performance criteria analyzed. It failed to solve only 31 instances, MIP failed to solve 57 instances, and T&B had 63 unsolved instances. ICBD obtained the lowest overall average gap of 0.07%, while the T&B and MIP methods obtained 0.17% and 0.28%, respectively. In all instance groups, ICBD obtained an average gap that was lower than or equal to the other methods. The instances with 60 jobs and 5 machines obtained the highest gaps: the MIP, T&B, and ICBD methods obtained 3.18%, 1.08%, and 0.68%, respectively.The average execution time of ICBD was 377.86 s, while those of the T&B and MIP methods were 573.89 s and 706.28 s, respectively. A decrease of 51.88% at runtime with ICBD method over the T&B method, higher than obtained using the instances from Tran and Beck [8], because the instances from Vallada and Ruiz [11] need more iterations to be solved, then the proposed improvement methods obtain better results. In the T&B method, 7.67% and 15.17% of the instances are solved with one and two iterations, respectively, much lower percentages than using instances from Tran and Beck [8]. The results indicate that Vallada and Ruiz instances are more difficult to obtain the optimal solution. Therefore, justifies the use of the three improvement procedures.

Table 3 – Comparison of the MIP, T&B, and ICBD methods based on the number of unsolved instances, average gap, and execution time.

n	m	# Uns				% Gap			Time		
		MIP	T&B	ICBD		MIP	T&B	ICBD	MIP	T&B	ICBD
20	2	0	0	0		0	0	0	0.95	*0.89*	1.07
	3	0	0	0		0	0	0	2.52	2.14	*1.87*
	4	0	0	0		0	0	0	8.17	9.01	*4.27*
	5	0	0	0		0	0	0	31.12	31.05	*11.65*
30	2	0	0	0		0	0	0	3.48	*2.79*	3.06
	3	0	0	0		0	0	0	17.27	34.62	*10.99*
	4	*0*	1	*0*		*0*	0.01	*0*	145.81	233.14	*83.26*
	5	0	0	0		0	0	0	365.14	287.38	*85.61*
40	2	0	0	0		0	0	0	10.85	*3.41*	4.93
	3	0	0	0		0	0	0	66.68	30.9	*16.09*
	4	0	0	0		0	0	0	466.00	348.09	*111.52*
	5	*2*	8	*2*		0.15	0.67	*0.05*	1463.93	1446.89	*743.84*
50	2	0	0	0		0	0	0	40.54	*5.69*	9.96
	3	*0*	1	*0*		0	0.01	0	257.40	280.30	*103.89*
	4	3	4	*1*		0.11	0.11	*0.02*	1545.20	1212.69	648.19
	5	18	16	*11*		1.87	1.22	*0.53*	2993.74	2542.33	2031.87
60	2	0	0	0		0	0	0	82.51	24.34	*19.72*
	3	*1*	3	*1*		0.02	0.03	*0.01*	844.00	737.04	*247.40*
	4	7	9	*3*		0.29	0.28	*0.06*	2306.63	1446.28	*929.21*
	5	26	21	*13*		3.18	1.08	*0.68*	3473.69	2798.76	*2488.82*
	Sum	57	63	*31*	Average	0.28	0.17	*0.07*	706.28	573.89	*377.86*

The average number of iterations using the original master problem is 1.87 for ICBD and 8.90 for T&B, and this difference is due in part to the average number of iterations performed by the warm-start procedure, which is 6.63. Adding together the number of both iterations, the ICBD method uses on average 8.5 iterations. Although both methods have almost the same number of iterations, the iterations during the warm-start procedure consume less computational time than those of the original master problem, and since there are more of them in ICBD, it makes this method faster. The quantity of CBCs generated by the ICBD during the warm-start procedure (56.76) is higher than those generated during the execution of the original master problem (10.07). The ICBD produces on average 66.83 CBCs in both phases, much more than the T&B method, which produces an average of 32.40 CBCs. This is because of the multi-cut procedure. The early termination of the master problem occurs on average 1.89 times in all instances; however, as the number of jobs increases, the number of times that this procedure is executed also increases. For example, in the instances with 60 jobs and 3 machines, it occurs on average 4.60 times. For ICBD, the average run time of the warm-start procedure is 193.18 s, which is greater than the average run time of original master problem, which is 184.68 s. The sum of these two run times is 377.86 s, which is less than the average run time of the T&B method (573.89 s). These results are shown in Table 4, where columns 1 and 2 refer to the number of jobs and machines, respectively. The average numbers of iterations (#iter) and cuts (#cut) during the execution of the original master problem were measured for both the T&B and ICBD methods. In addition, the average numbers of master problem terminations (#ta), warm-start iterations (#ws iter), and CBCs generated in the warm-start procedure (#ws cut) were measured for ICBD. The average ICBD execution times of the warm-start procedure (ws time) and original master problem (original time) were also measured.

Table 4 – Comparison of T&B and ICBD relative to traditional and proposed convergence acceleration elements.

n	m	T&B		ICBD						
		# iter	# cut	# ta	# ws iter	# iter	# ws cut	# cut	ws time	normal time
20	2	3.73	7.47	0.30	4.27	1.47	12.93	4.13	0.74	0.33
	3	4.53	13.60	0.60	5.20	1.40	24.10	5.60	1.13	0.75
	4	5.17	20.67	0.53	5.00	1.30	36.40	6.53	2.33	1.93
	5	5.37	26.83	0.47	4.80	1.13	44.67	6.33	5.22	6.44
30	2	5.40	10.80	0.53	5.17	2.07	15.47	6.53	1.89	1.18
	3	9.77	29.30	1.33	6.63	1.80	40.60	8.40	4.71	6.28
	4	9.77	39.07	1.57	6.43	1.63	60.40	10.40	23.96	59.3
	5	7.93	39.67	0.90	5.67	1.23	62.17	7.50	50.54	35.07
40	2	4.73	9.47	0.47	4.77	2.00	14.87	6.87	3.09	1.84
	3	8.50	25.50	1.63	6.47	1.43	39.50	5.70	8.34	7.75
	4	12.23	48.93	1.77	7.87	1.33	76.13	7.20	65.99	45.53
	5	9.17	45.83	1.97	6.97	1.47	89.83	10.50	374.79	369.05
50	2	5.07	10.13	0.90	5.17	2.30	16.67	7.53	6.56	3.4
	3	14.27	42.80	3.73	7.30	3.90	48.10	22.50	20.61	83.28
	4	13.10	52.40	2.90	8.93	2.00	92.40	12.13	345.19	303.00
	5	11.43	57.17	3.57	8.47	1.57	130.50	14.50	1083.98	947.89
60	2	7.70	15.40	1.53	5.80	2.70	20.20	10.40	11.34	8.39
	3	16.97	50.90	4.60	9.03	3.57	69.00	23.50	58.43	188.97
	4	14.03	56.13	4.53	10.40	1.73	114.67	12.40	485.55	443.66
	5	9.17	45.83	4.00	8.30	1.33	126.67	12.67	1309.28	1179.53
Average		8.90	32.40	1.89	6.63	1.87	56.76	10.07	193.18	184.68

5. Conclusions

The master problem of Benders decomposition provides a tight LB, as the optimality gap after the first iteration is at most 5% of the UB. Hence, the difficulty of the method is that there may be many solutions in the master problem that are smaller than the optimal solution of the original problem. Until all these solutions are found and evaluated by the subproblem, the method cannot be terminated with a gap of 0%. Therefore, the challenge is to find these solutions as quickly as possible. With that in mind, the proposed procedures seek to quickly find them. These procedures consist of an early termination of the master problem execution when a repeated LB is found, a multi-cut procedure that evaluates more than one solution at a time, and finally, a warm-start procedure, in which quality solutions are found more quickly. No procedure was developed to accelerate the subproblem solutions because they consume much less computational time than the master problem.

The proposed acceleration procedures have not before been applied to the UPMSP-SMDST. Furthermore, they can be used with a combinatorial Benders decomposition in any other problem. In addition, the results show that the procedures improve the performance of the Benders decomposition scheme of Tran and Beck [8]. Moreover, the proposed method also performed better than the mixed-integer formulation of Avalos-Rosales et al. [1] relative to the three performance criteria analyzed.

The approach used by Geoffrion and Graves [32] of solving the master problem only once and generating a Benders cut each time a better incumbent solution is found was tested and used more computational time than the traditional approach. One hypothesis why this happened is that to implement this procedure, it is necessary to use a CPLEX callback function that disables the dynamic search used to improve CPLEX performance. The local branching strategy of Fischetti and Lodi [34] was also tested, but it consumed more computational time to find the repeated solutions of the proposed procedures.

One proposal for future work is to develop a strong cut that eliminates more solutions than just the infeasible solutions as in the no-good cut. Another proposal is to develop a heuristic to create quality cuts to be inserted into the master problem before starting the Benders decomposition procedures themselves, as in Sherali and Lunday [24].

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgements

The authors would like to acknowledge Dr. Tony T. Tran for providing the instances used in Tran e Beck [8], and also the CNPq (National Council for Scientific and Technological Development), CAPES (Coordination of Personnel Improvement of Higher Education), and FAPEMIG (Foundation for Research Support of the State of Minas Gerais) for financial support.

References

1. O. Avalos-Rosales, F. Angel-Bello, A. M. Alvarez, "Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times," *The International Journal of Advanced Manufacturing Tecnology*. vol.76, no. 9, pp. 1705-1718, 2015.

2. M. Afzalirad, M. Shafipour, "Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions," *Journal of Intelligent Manufacturing* (2015), doi:10.1007/s10845-015-1117-6.

3. E. Mokotoff, "Parallel machine scheduling problems: a survey," *Asia-Pacific Journal of Operational Research*, vol. 18, no. 2, pp. 193–242, 2001.

4. A. Allahverdi, H. M. Soroush, "The significance of reducing setup times/setup costs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 978–984, 2008.

5. J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, J. Weglarz, (1996). "Scheduling computer and manufacturing processes". Berlin: Springer.

6. P. Rocha, M. G. Ravetti, G. R. Mateus, P. M. Pardalos, "Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times," *Computers & Operations Research*, vol. 35, no. 4, pp. 1250–1264, 2008.

7. M. R. Paula, G. R. Mateus, M. G. Ravetti, "A non-delayed relax-and-cut algorithm for scheduling problems with parallel machines, due dates and sequence-dependent setup times," *Computers & Operations Research*, vol. 37, no. 5, pp. 938–949, 2010.

8. T. Tran, J. C. Beck, (2012). “Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups”. In Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012), 774–779.

9. M. R. Paula, M. G. Ravetti, G. R. Mateus, P. M. Pardalos, “Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search,” *IMA Journal of Management Mathematics*, vol. 18, no. 2, pp. 101–115, 2007.

10. S.-W Lin, C.-C. Lu, K.-C Ying, “Minimization of total tardiness on unrelated parallel machines with sequence-and machine-dependent setup times under due date constraints,” *The International Journal of Advanced Manufacturing Technology*, vol. 53, no. 1, pp. 353–361, 2011.

11. E. Vallada, R. Ruiz, “A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times,” *European Journal of Operational Research*, vol. 211, no. 3, pp. 612–622, 2011.

12. K. Ying, Z. Lee, S. Lin, “Makespan minimization for scheduling unrelated parallel machines with setup times,” *Journal of Intelligent Manufacturing*, vol. 23, no. 5, pp. 1795–1803, 2012.

13. J.-H Lee, J.-M Yu, D.-H Lee, “A tabu search algorithm for unrelated parallel machine scheduling with sequence-and machine-dependent setups: minimizing total tardiness,” *The International Journal of Advanced Manufacturing Technology*, vol. 69, no. 9, pp. 2081–2089, 2013.

14. J.-P. Arnaout, R. Musa, G. Rabadi, “A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines—part II: Enhancements and experimentations,” *Journal of Intelligent Manufacturing*, vol. 25, no. 1, pp. 43–53, 2014.

15. J. N. Hooker, “A hybrid method for planning and scheduling,” *Constraints*, vol. 10, no. 4, pp. 385–401, 2005.

16. J. N. Hooker, “An integrated method for planning and scheduling to minimize tardiness,” *Constraints*, vol. 11, no. 2, pp. 139–157, 2006.

17. J. N. Hooker, “Planning and scheduling by logic-based Benders decomposition,” *Operations Research*, vol. 55, no. 3, pp. 588–602, 2007.

18. H. Li, K. Womer, “Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm,” *Journal of Scheduling*, vol. 12, no. 3, pp. 281-298, 2009.

19. E. Coban, J. N. Hooker, “Single-facility scheduling by logic-based Benders decomposition,” *Annals of Operations Research*, vol. 210, no. 1, pp. 245-272, 2013.

20. J. F. Benders, “Partitioning procedures for solving mixed-variables programming problems,” *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

21. J. N. Hooker, G. Ottosson, “Logic-based Benders decomposition,” *Mathematical Programming*, vol. 96, no. 1, pp. 33–60, 2003.

22. R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, “Optimization and approximation in deterministic sequencing and scheduling: A survey,” *Annals of Discrete Mathematics*, vol. 5, no. 2, pp. 287–326, 1979.

23. G. K. D. Saharidis, M. G. Ierapetritou, “Improving Benders decomposition using maximum feasible subsystem (MFS) cut generation strategy,” *Computers and Chemical Engineering*, vol. 8, no. 34, pp. 1237–1245, 2010.

24. H. D. Sherali, B. J. Lunday, “On generating maximal non dominated Benders cuts,” *Annals of Operations Research*, vol. 210, no. 1, pp. 57-72, 2013.

25. N. Papadakos, “Practical enhancement to the Magnanti–Wang method,” *Operations Research Letters*, vol. 36, no. 4, pp. 444–449, 2008.

26. W. Rei, M. Gendreau, J. F. Cordeau, P. Soriano, “Accelerating Benders decomposition by local branching,” *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 333–345, 2009.

27. G. K. D.Saharidis, M. Minoux, M. G. Ierapetritou, “Accelerating Benders method using covering cut bundle generation,” *International Transactions in Operational Research*, vol. 17, no. 2, pp. 221–237, 2010.

28. G. K. D. Saharidis, M. G. Ierapetritou, “Speed-up Benders decomposition using maximum density cut (MDC) generation,” *Annals of Operations Research*, vol. 210, no. 1, pp. 101–123, 2013.

29. N. Azad, G. K. D.Saharidis, H. Davoudpour, H. Malekly, S. A. Yektamaram, “Strategies for protecting supply chain networks against facility and transportation disruptions: An improved Benders decomposition approach,” *Annals of Operations Research*, vol. 210, no. 1, pp. 125–163, 2013.

30. D. McDaniel, M. Devine, “A modified Benders partitioning algorithm for mixed integer programming,” *Management Science*, vol. 24, no. 3, pp. 312–319, 1977.

31. D. Wheatley, F. Gzara, E. Jewkes, “Logic-based Benders decomposition for an inventory-location problem with service constraints,” *Omega*, vol. 55, pp. 10–23, 2015.

32. A. M. Geoffrion, G. W. Graves, “Multicommodity distribution system design by benders decomposition,” *Management Science*, vol. 20, no. 5, pp. 822–844, 1974.

33. G. Cote, M. A. Laughton, “Large-scale mixed integer programming: benders-type heuristics,” *European Journal of Operational Research*, vol. 16, no. 3, pp. 327–333, 1984.

34. M. Fischetti, A. Lodi, “Local branching,” *Mathematical Programming*, vol. 98, no 1–3, pp. 23–47, 2003.

35. C. A. Poojari, J. E. Beasley, “Improving Benders decomposition using a genetic algorithm,” *European Journal of Operational Research*, vol. 199, no. 1, pp. 89-97, 2009.

36. Z. Huang, Q. P. Zheng, “Decomposition-based exact algorithms for risk-constrained traveling salesman problems with discrete random arc costs,” *Optimization Letters*, vol. 9, no. 8, pp. 1553–1568, 2015.

37. H. D. Sherali, K.-H. Bae, M. Haouari, “A Benders decomposition approach for an integrated airline schedule design and fleet assignment problem with flight retiming, schedule balance, and demand recapture,” *Annals of Operations Research*, vol. 210, no. 1, pp. 213-244, 2013.

38. M. Jenabi, S. M. T. Fatemi Ghomi, S. A. Torabi, S. H. Hosseinian, “Acceleration strategies of Benders decomposition for the security constraints power system expansion planning,” *Annals of Operations Research*, vol. 235, no. 1, pp. 337-369, 2015.

39. F. You, I. E. Grossmann, “Multicut Benders decomposition algorithm for process supply chain planning under uncertainty,” *Annals of Operations Research*, vol. 210, no. 1, pp. 191-211, 2013.

40. T. Magnanti, R. Wong, “Accelerating Benders decomposition algorithmic enhancement and model selection criteria,” *Operations Research*, vol. 29, no. 3, pp. 464–484, 1981.

41. V. Jain, I. Grossmann, “Algorithms for hybrid MILP/CP models for a class of optimization problems,” *INFORMS Journal on Computing*, vol. 13, no. 4, pp. 258–276, 2001.

42. L. A. Hall, A. S. Schulz, D. B. Shmoys, J. Wein, “Scheduling to minimize average completion time: off-line and on-line approximation algorithms,” *Mathematics of Operations Research*, vol. 22, no. 3, pp. 513-544, 1997.

43. C. Phillips, C. Stein, J. Wein, “Minimizing average completion time in the presence of release dates,” *Mathematical Programming*, vol. 82, no. 1, pp. 199-223, 1998.

44. J. M. Van den Akker, C. P. M. Van Hoesel, M. W. P. Savelsbergh, “A polyhedral approach to single-machine scheduling problems,” *Mathematical Programming*, vol. 85, no. 3, pp. 541-572, 1999.

45. L. Fanjul-Peyro, R. Ruiz, “Size-reduction heuristics for the unrelated parallel machines scheduling problem,” *Computers & Operations Research*, vol. 38, no. 1, pp. 301–309, 2011.

Appendix B

Mathematical Formulation and Hybrid Column Generation for Minimizing Total Tardiness in a Scheduling Problem with Unrelated Parallel Machines

Francisco Regis Abreu Gomes[1,*], Geraldo Robson Mateus[2]

[1]Graduate Program in Production Engineering, Federal University of Minas Gerais, Brazil and Federal Institute of Education, Science and Technology of Ceará, Brazil

[2] Computer Science Department, Federal University of Minas Gerais, Brazil

* Corresponding author. Tel.: +55 85 98873 4453.
E-mail address: regisgomes@ifce.edu.br (Regis Gomes).

Abstract

This paper addresses the NP-hard unrelated parallel machine scheduling problem with sequence and machine-dependent setup times for minimizing total tardiness. Mathematical models for this problem often use a constant known as big-M on account of the disjunctive constraints. This yields very weak lower bounds that make it difficult to obtain the optimal solution, even for small-size instances. To address this problem, we propose a mathematical formulation that does not use the big-M constant. To this end, we present an approach that uses dummy jobs instead of the big-M constant. Additionally, an optimality condition method that reduces the solution space of the problem is proposed. Experiments conducted on five instance types produced computational proof of the superiority of the proposed model compared to models based on Wagner's (1959) and Manne's (1960) formulations. The proposed model produced 291 optimal solutions compared to 98 and 148 of Wagner's (1959) and Manne's (1960) models, respectively, and it was up to three orders of magnitude faster in the 300 small-size instances that were tested. A column-generation algorithm is also proposed to find near-optimal solutions for medium-size instances with up to 50 jobs and ten machines. Unlike standard approaches, the proposed model is used instead of a dynamic programming algorithm to solve the pricing problem. For accelerating the convergence of the column-generation algorithm, various heuristics are proposed to generate the initial columns and solve the pricing problem. The hybrid column generation obtained an average gap and runtime of 2.71% and 930.48 s, respectively, compared to 34.78% and 2,490.37 s, respectively, of the proposed model. Results indicate that the proposed approaches are more effective in terms of both running time and solution quality. In addition, these approaches can be used in other scheduling problems that use the big-M constant.

Keywords: scheduling; unrelated parallel machines; column generation; tardiness; heuristics.

1. Introduction

In today's competitive business environment of manufacturing and services, efficient scheduling is one of the most critical issues (Afzalirad & Shafipour, 2015). The parallel machine scheduling problem (PMSP) is broadly applied in many manufacturing and service systems. It has therefore been a subject of continuing interest for researchers and practitioners (Mokotoff, 2001). Many types of PMSPs have been proposed in the literature. They can be classified into identical, uniform, and unrelated parallel machine categories (Cheng & Sin, 1990). Of these types, the unrelated PMSP, which includes the machine and sequence-dependent setup times and total tardiness as criteria, has received less attention than other PMSPs (Lee, Yu, & Lee, 2013). However, with the adoption by companies of the just-in-time philosophy, an increasing amount of research in the past two decades has involved tardiness (Lin, Chou, &Ying, 2007). Nevertheless, tardiness is a difficult criterion with which to work, even in the single-machine environment (Lin & Ying, 2007). Applications of all PMSP types are common in many industries, including painting, plastic, textile, glass, semiconductor, chemical, and paper manufacturing (Chang & Chen, 2011).

Exact mathematical programming approaches for scheduling problems use two distinct types of formulations (Pessoa, Uchoa, Aragão, & Rodrigues, 2010): (1) formulations whereby the job sequence is represented by binary variables and completion times are denoted by continuous variables; and (2) time-indexed formulations, whereby the completion time of each job is represented by binary variables indexed over a discrete time horizon (Sousa & Wolsey, 1992). The formulations of type (2) are known to yield tight linear relaxations; however, they cannot be directly applied to many instances on account of their pseudo-polynomially large number of variables. The formulations of type (1) are compact in that they involve a polynomial number of variables and constraints. On the other hand, they yield poor linear relaxations. This is notoriously due to the big-M constant having to linearize the disjunctive constraints (Koné, Artigues, Lopez, & Mongeau, 2013). The formulations of type (2), on the other hand, do not use this constant.

Avalos-Rosales, Angel-Bello, and Alvarez (2015) proposed several mixed integer formulations of type (1) for a PMSP to minimize the makespan. These formulations outperform the previously published formulations in terms of the instance size and computational time for reaching optimal solutions. Using these models, it is possible to solve instances up to 60 jobs and five machines that are six times larger than was previously solved. In addition, they enable attainment of optimal solutions for instances of the same size up to four orders of magnitude faster. This is only possible because those authors proposed a linearization method to calculate a makespan that does not use the big-M constant. We emphasize that these formulations still use this constant in the disjunctive constraints. Unfortunately, these formulations thus cannot be used when the criterion is the minimization of total tardiness once the new linearization applies only to computing the makespan. To the best of our knowledge, a formulation of type (1) for PMSPs with tardiness as a criterion that does not use the big-M constant does not exist.

Inspired by the performance achieved by the formulation of Avalos-Rosales, Angel-Bello, and Alvarez (2015) we propose a mathematical formulation for the problem under study that does not use the big-M constant. To this end, we employ dummy jobs instead of the big-M constant to linearize the computation of the total tardiness of the jobs. We additionally propose an optimality condition that reduces the solution space of the problem. Computational results showed that the proposed model obtained tight linear relaxations, more optimal solutions, the best feasible solutions, and smaller runtimes than existing models in the literature. These are the first contributions of this paper.

For larger instances, the proposed model has difficulty obtaining the optimal solution, especially for instances with looser due dates. Therefore, an approach based on column generation (CG) combined with some heuristics is proposed. CG is an optimization method to solve a variety of combinatorial optimization problems, such as hybrid flow shop scheduling (Figielska, 2009), job shop scheduling (Jampani & Mason, 2010), server cluster optimization (Kramer, Petrucci, Subramanian, & Uchoa, 2012), roll-on/roll-off routing (Hauge, Larsen, Lusby, & Krapper, 2014), and vehicle scheduling (Guedes & Borenstein, 2015). The theoretical basis of the CG algorithm has been provided by Dantzig and Wolfe (1960). The linear program is divided into two problems, which are referred to as the relaxed and restricted master problem and subproblem. The subproblem uses the dual information from the master problem to generate new columns that can potentially improve the objective function of the master problem.

The column in our work corresponds to a job sequence. The iterative process of generating new columns using the subproblem and adding them to the master problem terminates when the subproblem generates new columns with non-negative reduced costs. Usually, it starts from an initial set of columns. This approach avoids the difficulty of explicitly generating all columns of the problem. This approach was suggested by Gilmore and Gomory (1961) for solving cutting stock problems. Later, the integrality constraints of the master problem are introduced and solved as a mixed integer program.

Usually in PMSPs addressed by CG, the subproblems have some structural dominance property that enables solving by a pseudo-polynomial dynamic programming algorithm in reasonable time. However, in the PMSP under this study, any structural dominance property is identified. Therefore, the proposed model is used to solve the subproblems. The quality of the hybrid CG algorithm is because of the efficiency of the proposed model in solving the subproblems. This quality is also engendered by the proposed heuristics for the initial solution and for solving the subproblems. These comprise the other contributions of this paper.

The remainder of this paper is organized as follows. Section 2 reviews the solution approaches for PMSPs and CG applications to the scheduling problems. Section 3 presents two mathematical models from the literature and a new mathematical formulation is proposed. In Section 4, a new hybrid column generation algorithm is additionally proposed. Section 5 describes the computational experiments comparing the mathematical formulations from the literature, the proposed formulation, and the hybrid CG approaches, and the results are reported. In Section 6, the conclusions are presented.

2. Literature review

This section reviews the previous studies on applying PMSPs and CG to the above-described problems. Our review is restricted to PMSPs with the due date and setup times because these features are considered in this paper. For details on parallel machine scheduling problems, the due date as a criterion, and the setup time in scheduling problems, see Li and Yang (2009), Vallada, Ruiz, and Minella (2008), and Allahverdi, Ng, Cheng, and Kovalyov (2008), respectively.

Most previous studies have been conducted on identical or uniform PMSPs only with sequence-dependent setup times (Lee, Yu, & Lee, 2013). Lee and Pinedo (1997) suggest a three-phase heuristic using the apparent tardiness cost with setups (ATCS) rule, a dispatching rule, and a simulated annealing algorithm for minimizing the sum of the weighted tardiness. For minimizing the total tardiness, Park, Kim, and Lee (2000) improve the dispatching rule using look-ahead parameters calculated by a neural network. Bilge, Kirac, Kurtulan, and Pekgun (2004) propose a tabu search approach, whereby the candidate list strategies, tabu classifications, tabu tenures, and intensification/diversification strategies are investigated. Anghinolfi and Paolucci (2007) propose a hybrid metaheuristic that incorporates the core features of simulated annealing, the tabu search, and the variable neighborhood search. Armentano and de França Filho (2007) propose GRASP(*Greedy* Randomized Adaptive Search Procedure)-based search heuristics that incorporate adaptive memory principles.

For the unrelated PMSP with only sequence-dependent setup times, Kim, Kim, Jang, and Chen (2002) suggest a simulated annealing algorithm with various interchange and insertion methods for minimizing the total tardiness. For minimizing the weighted number of tardy jobs, M'Hallah and Bulfin (2005) propose branch and bound algorithms, while Chen and Chen (2009) propose hybrid metaheuristics that integrate the tabu search and variable neighborhood descent approach. In addition, Chen (2012) presents several iterated hybrid metaheuristic algorithms, while Zhu and Heady (2000) propose a mixed integer programming model to minimize the sum of earliness and tardiness penalties.

For the unrelated PMSP with the machine and sequence-dependent setup time, few studies have been performed. For minimizing the total tardiness, Chen (2009) considers the problem with an additional strict due date constraint for some jobs. That author proposes a simulated annealing algorithm that incorporates the feasibility improvement method. In addition, Lin, Lu, and Ying (2011) propose an iterated greedy algorithm and a simple dispatching rule, which are respectively referred to as primary customers and the shortest completion time, to generate the initial solution. Lee, Yu, and Lee (2013) propose a tabu search algorithm that incorporates various neighborhood generation methods. It showed better performance than the two previous methods.

Meanwhile, Rocha, Ravetti, Mateus, and Pardalos (2008) propose a branch and bound algorithm and a GRASP metaheuristic for minimizing the makespan added to the weighted tardiness. Paula, Mateus, and Ravetti (2010) propose a non-delayed relax-and-cut algorithm based on the Lagrangian relaxation of a time-indexed formulation to minimize the total weighted tardiness. For minimizing the total earliness and tardiness penalties, Nogueira et al. (2014) propose three different heuristics based on the GRASP metaheuristic, and Zeidi and Hosseini (2015) propose a genetic algorithm with a simulated annealing method as a local search procedure to improve the solution quality.

Most previous studies that have employed CG address identical PMSPs; however, they do not consider the setup times. Van den Akker, Hoogeveen, and van de Velde (1999) efficiently solved the linear programming relaxation problem of minimizing the total weighted completion time using CG with identical parallel machines. Their approach is formulated as a set covering problem, and the pricing problem is solved in a pseudo-polynomial time. Chen and Powell (1999a) addressed several PMSPs considering identical, uniform, and unrelated machines with two objectives: to minimize the total weighted completion time, and to minimize the weighted number of tardy jobs. Chen and Powell (1999b) addressed the identical PMSP with an unrestrictive large common due date to minimize the total weighted earliness and tardiness. Moreover, Chen and Lee (2002) addressed the identical parallel machine problem with a common due date window to minimize the total weighted earliness and tardiness. Chen and Powell (2003) addressed the multiple job family scheduling problem on identical parallel machines. They employed sequence-dependent or sequence-independent setup times to minimize the total weighted completion time of the jobs.

The above four studies developed exact methods based on branch and bound solving at each node by CG, while the subproblems are solved by pseudo-polynomial dynamic programming algorithms. According to Lopes and de Carvalho (2007), the success of previous work was only possible because their versions of PMSPs have structural dominance properties (usually corresponding to some job-ordering restrictions) that enable reducing the subproblem solution space. Consequently, the solution of larger instances is made possible. Lopes and de Carvalho (2007) addressed the unrelated PMSP with sequence-dependent setup times and availability dates for the machine and release dates for the jobs. Nevertheless, the authors failed to identify any structural dominance property. To overcome this drawback, they proposed a method to reduce the search space to accelerate the solving from a two-cycle elimination version of the dynamic programming algorithm. Their approach is based on the procedure proposed by Houck, Picard, Queyranne, and Vemuganti (1980). Finally, they used it in a branch-and-price algorithm for minimizing the weighted tardiness.

Furthermore, van den Akker, Hoogeveen, and van Kempen (2012) used a destructive strategy to compute a lower bound and CG to compute an upper bound on the number of machines necessary to feasibly accommodate all jobs of identical PMSP. Accordingly, the maximum lateness can be minimized, albeit subject to release dates, deadlines, and/or generalized precedence constraints. To the best of our knowledge, identifying the exact PMSP solution of the problem studied not yet been investigated. This is likely on account of the difficulty of solving the related subproblem.

3. Parallel machine scheduling problem for minimizing total tardiness

3.1 Problem description

The scheduling problem investigated in this study considers n independent jobs, $J = \{1, 2, . . . , n\}$, on m unrelated parallel machines, $I = \{1, 2, . . . , m\}$. Each machine $i \in I$ is ready at time zero and can process all jobs. Each job $j \in J$ is processed by exactly one of the machines, has m processing times p_{ij} ($i \in I$), is available in time zero, and has a due date d_j. A machine and sequence-dependent setup time, s_{ijl}, is incurred between two different jobs, $j \neq l$. The machine setup can be started and completed during the idle time, as commonly assumed in the literature (Potts & Kovalyov, 2000). All the parameters are deterministic non-negative integers. A job sequence is a subset of J processed by the machine in a sequence, in which each job is non-preemptively processed only once. Each job in a sequence has a completion time, C_j, and tardiness is defined as $T_j = \max\{0, C_j - d_j\}$. The aim is to find the set of job sequences that processes all jobs and minimizes their total tardiness. In the standard three-field notation, this problem is denoted as R/s_{ijl}, $d_j/\Sigma T_j$. It is NP-hard because it is an extension of the NP-hard $1/d_j/\Sigma T_j$ (Lawler, 1977).

3.2 Wagner model

Rocha, Ravetti, Mateus, and Pardalos (2008) adapted for the parallel machine scheduling problem the models based on sequence-position variables proposed by Wagner (1959) and precedence variables proposed by Manne (1960), both of which were originally proposed for the job shop problem. In the Wagner model, α_{ijp} is one if job j is processed in machine i in the p-th position (and zero, otherwise), β_{ijlp} is one if jobs j and l are processed by machine i at the p-th and (p + 1)-th positions, respectively, (and zero, otherwise), and t_{ip} denotes the starting time in machine i in the p-th position. In this model, the position amount p is equal to the number of jobs. The model itself is the following:

$$\min \sum_{j \in J} T_j \qquad (1)$$

$Subject\ to$:

$$\sum_{i \in I} \sum_{p \in J} \alpha_{ijp} = 1, \qquad \forall j \in J \qquad (2)$$

$$\sum_{j \in J} \alpha_{ijp} \leq 1, \qquad \forall\, i \in I, p \in J \qquad (3)$$

$$\sum_{j \in J} \alpha_{ijp} \leq \sum_{l \in J} \alpha_{il(p-1)}, \qquad \forall i \in I, p \in J, p \geq 2 \qquad (4)$$

$$\beta_{ijl(p-1)} + 1 \geq \alpha_{ij(p-1)} + \alpha_{ilp}, \qquad \forall\, j, l, p \in J, j \neq l, p \geq 2, i \in I \qquad (5)$$

$$t_{ip} \geq t_{i(p-1)} + \sum_{j \in J} p_{ij}\, \alpha_{ij(p-1)} + \sum_{j \in J} \sum_{\substack{l \in J \\ l \neq j}} s_{ijl} \beta_{ijl(p-1)}, \qquad \forall\, i \in I, p \in J, p \geq 2 \qquad (6)$$

$$T_j \geq t_{ip} + p_{ij} - \left(1 - \alpha_{ijp}\right)M - d_j, \qquad \forall\, j, p \in J, i \in I, \qquad (7)$$

$$t_{ip} \geq 0, \qquad \forall\, p \in J \qquad (8)$$

$$\alpha_{ijp} \in \{0,1\}, \qquad \forall j, p \in J, i \in I, \qquad (9)$$

$$\beta_{ijlp} \in \{0,1\}, \qquad \forall j, l, p \in J, j \neq l, i \in I \qquad (10)$$

The objective function (1) minimizes the total tardiness of the jobs. Constraints (2) ensure that each job is assigned to only one machine and one position. Constraints (3) ensure that no more than one job is assigned to each position of a machine. Constraints (4) ensure that if a job is assigned to a position p, $p \geq 2$, another job is assigned to position $p - 1$ of the same machine. Constraints (5) determine the sequence of jobs on the machines. Constraints (6) calculate the start time of the positions on each machine. Constraints (7) calculate the tardiness of each job. Finally, the constraints (8) to (10) define the conditions of non-negativity and integrality of the variables.

3.3 Manne model

In the Manne model, α_{ij} is one if job j is processed in machine i (and zero, otherwise), β_{ijl} is one if the job l is processed after (not necessarily immediately after) job j in machine i (and zero, otherwise), and t_j denotes the starting time of job j. The model itself is the following:

$$\min \sum_{j \in J} T_j \qquad (11)$$

$Subject\ to$:

$$\sum_{i \in I} \alpha_{ij} = 1, \qquad \forall j \in J \qquad (12)$$

$$(2 - \alpha_{ij} - \alpha_{il})M + (1 - \beta_{ilj})M + t_j \geq t_l + p_{il} + s_{ilj}, \qquad \forall j, l \in J, j \neq l, i \in I \qquad (13)$$

$$(2 - \alpha_{ij} - \alpha_{il})M + \beta_{ilj}M + t_l \geq t_j + p_{ij} + s_{ijl}, \qquad \forall j, l \in J, j \neq l, i \in I \qquad (14)$$

$$T_j \geq \left(t_j + \sum_{i \in I} p_{ij}\alpha_{ij}\right) - d_j, \qquad \forall j \in J \qquad (15)$$

$$t_j \geq 0, \qquad \forall j \in J \qquad (16)$$

$$T_j \geq 0, \qquad \forall j \in J \qquad (17)$$

$$\alpha_{ij} \in \{0,1\}, \qquad \forall j \in J, i \in I \qquad (18)$$

$$\beta_{ijl} \in \{0,1\}, \qquad \forall j \in J, l \in J, j \neq l, i \in I \qquad (19)$$

The objective function (11) minimizes the total tardiness of the jobs. Constraints (12) ensure that each job is processed by only one machine. Constraints (13) and (14) describe the precedence relationship between the jobs, i.e., for each pair of jobs, (l, j) or j is processed after l, or l is processed after j. Constraints (15) calculate the tardiness of each job. Finally, constraints (16) to (19) define the non-negativity and integrality of the variables.

3.4 Positional model

The proposed model uses the same type of positional variable proposed by Wagner (1959). Hence, it was given the "positional model" name. In this Wagner model, the number of positions per machine is equal to the number of jobs of the problem. The big-M constant is used to determine which of the available positions is occupied. In practice, only a portion of the positions is occupied by jobs. In the positional model, the positions not occupied by jobs (called "real jobs") are now occupied by a job created exclusively for this purpose, called the "dummy job." Thus, all positions are occupied by real or dummy jobs.

The dummy job is represented by zero. The real jobs are allocated to only one position of a machine. The dummy job can be allocated to no position of a machine or to more than one. The dummy job does not affect the objective function value of the problem; thus, its parameters d_0, p_{i0}, and s_{i0j} must have values equals to zero, and parameters s_{ij0} have large values. Therefore, the dummy job is allocated in the first position, and the real jobs are allocated after the dummy job ($S_1 = \{0, j_1, j_2, ...\}$). In this case, the setup time that occurs is s_{i0j}, which is equal to zero. Consequently, it does not affect the value of the objective function. If the dummy job is allocated between real jobs ($S_2 = \{ j_1,..., 0, .., j_2, ...\}$), one of the setup times that occurs is s_{ij0}, which is a very large value. It is so large that it greatly increases the value of the objective function. Thus, the solution process is induced to place the dummy job before the real jobs and never between them.

The first innovation of the positional model in relation to the presented models is to not use the big-M constant to linearize the disjunctive constraints (or precedence constraints). Then, the model is originally linear and can therefore be quickly resolved (Chen & Powell, 1999a). The model has the following variables: x_{ijp} is one if job j is processed in machine i in the p-th position (and zero, otherwise), z_{ijlp} is one if jobs j and l are processed by machine i at the p-th and (p + 1)-th positions, respectively, (and zero, otherwise), C_{ip} is the completion time in the p-th position in machine i, and T_{ip} is tardiness in the p-th position in machine i.

The second innovation of the positional model is to use a number of positions per machine (k) that is smaller than the total number of real jobs. The aim is to make the most compact model. This is possible because, in practice, the number of jobs allocated by the machine is smaller than the total number of jobs. This is because the problem has more than one machine. This restricts the search space, which can eliminate the optimal solution. Therefore, an optimality condition must be developed in this case.

Proposition 1: The optimality condition defines that, if there is at least one dummy job allocated per machine in the job sequences of all machines, the optimal solution identified when $k < n$ is equal to the optimal solution when $k = n$.

Proof: Suppose there is an optimal solution for $k = n$. Let s^k be a set of optimal sequences for the PMSP with $k < n$, and the sequence of each machine contains a dummy job. In this case, any real job could be reallocated at any position or in place of a dummy job of some other sequence. If this new solution has a lower cost we have a contradiction, because s^k is an optimal solution. Therefore, s^k is really an optimal solution for $k < n$ and also for $k = n$. Suppose now there is a sequence in s^k, for any machine, with k real jobs and no dummy job. In this case, increasing the number of positions from k to k+1, there may be a real job that if reallocated to this sequence would generate a lower cost solution. Therefore, the optimality of s^k is not guaranteed for $k < n$, and it is necessary to increase the value of k until there is a dummy job for each sequence or until $k = n$.

It is not known in advance how many jobs will be allocated per machine. Therefore, the number of positions per machine should be adequate for all the real jobs allocated, and at least one dummy job is allocated per machine. In this study, we used the empirical formula: $k = \lceil n/m \rceil + 2$. The experiments indicated that the proposed formula for k always found the number of positions that met the optimality condition. However, if it fails, add one and one position and run the model again until the optimality condition be met. Then, the positional model considered p positions $P = \{1, 2, . . . , k\}$. The positional model is presented as follows.

$$Min \sum_{i \in I} \sum_{p \in P} T_{ip}, \qquad (20)$$

$Subject\ to$:

$$\sum_{i \in I} \sum_{p \in P} x_{ijp} = 1, \qquad \forall j \in J \qquad (21)$$

$$\sum_{j \in J+\{0\}} x_{ijp} = 1, \qquad \forall i \in I, p \in P \qquad (22)$$

$$\sum_{\substack{l \in J+\{0\} \\ j \neq l}} z_{ijlp+1} = x_{ijp}, \qquad \forall i \in I, j \in J+\{0\}, p \in P, p \leq k-1 \qquad (23)$$

$$\sum_{\substack{j \in J+\{0\} \\ j \neq l}} z_{ijlp} = x_{ilp}, \qquad \forall i \in I, l \in J+\{0\}, p \in P, p \geq 2 \qquad (24)$$

$$C_{i1} = \sum_{j \in J+\{0\}} p_{ij} x_{ij1}, \qquad \forall i \in I \qquad (25)$$

$$C_{ip} \geq C_{i,p-1} + \sum_{j \in J+\{0\}} \sum_{\substack{l \in J+\{0\} \\ j \neq l}} s_{ijl} z_{ijlp} + \sum_{l \in J+\{0\}} p_{il} x_{ilp}, \qquad \forall i \in I, p \in P, p \geq 2 \qquad (26)$$

$$T_{ip} \geq C_{ip} - \sum_{j \in J+\{0\}} d_j x_{ijp}, \qquad \forall i \in I, p \in P \qquad (27)$$

$$C_{ip} \geq 0, \qquad \forall i \in I, p \in P \qquad (28)$$

$$T_{ip} \geq 0, \qquad \forall i \in I, p \in P \qquad (29)$$

$$x_{ijp} \in \{0,1\}, \qquad \forall i \in I, j \in J+\{0\}, p \in P \qquad (30)$$

$$z_{ijlp} \in \{0,1\}, \qquad \forall i \in I, j, l \in J, j \neq l, p \in P, p \geq 2 \qquad (31)$$

The objective function (20) minimizes the total tardiness in all positions and, consequently, of all jobs. Constraints (21) ensure that each real job is processed by only one machine in one position. Constraints (22) ensure that all positions of all machines are occupied by only one real or dummy job. Constraints (23) and (24) describe the precedence relationships between jobs. That is, for each pair of jobs, (*l*, *j*) or *j* is processed immediately after *l*, or *l* is processed immediately after *j*. Constraints (25) calculate the completion time at the first position of each machine. Constraints (26) calculate the completion time at all positions except the first of each machine. Constraints (27) calculate the tardiness in the positions of each machine. Finally, constraints (28) to (31) define the non-negativity and integrality of the variables.

4. Column generation approach

4.1. Set partitioning problem formulation

The positional model is reformulated as a set partitioning problem via Dantzig–Wolfe decomposition. A column is a job sequence in a machine that satisfies constraints (21) to (31). Let

S_i be the set of feasible job sequences, s, for each machine, i, and y_{is} is the binary variable that is one if job sequence s is assigned for machine i (and zero, otherwise). For each job sequence $s \in S_i$, there is an associated cost, f_{is}, and a_{jis} is one if job j is covered by sequence s in machine i (and zero, otherwise). The reformulation of the positional model is the following:

$$Min \sum_{i \in I} \sum_{s \in S_i} f_{is} y_{is}, \tag{32}$$

$Subject\ to$:

$$\sum_{i \in I} \sum_{s \in S_i} a_{jis} y_{is} = 1, \qquad \forall j \in J \tag{33}$$

$$\sum_{s \in S_i} x_{is} = 1, \qquad \forall i \in I \tag{34}$$

$$x_{is} \in \{0,1\}, \qquad \forall i \in I, s \in S_i \tag{35}$$

The objective function (32) minimizes the sum of costs of chosen job sequences. Constraints (33) ensure that each job is covered by exactly one job sequence. Constraints (34) ensure that exactly one job sequence is assigned for each machine. Finally, constraints (35) define the binary variables.

The problem of (32) to (35) with a restricted number of columns $\overline{S_i} \subset S_i$ is the restricted master problem (RMP). In CG, the linear relaxation of the restricted master problem (RMP$_{LR}$) is solved by a linear programming method. Let π_j and θ_i be the dual variables for (33) and (34), respectively. Then, the reduced cost, r_{is}, of a job sequence $s \in S_i$ is given by:

$$r_{is} = f_{is} - \sum_{j \in J} \pi_j a_{jis} - \theta_i, \tag{36}$$

If there exists column s with a negative reduced cost, this column is added to the set, $\overline{S_i}$. The RMP$_{LR}$ is solved again to update the objective and dual variable values. If there is no column with a negative reduced cost, the current solution is optimal for the RMP$_{LR}$. The objective function value provides a lower bound for the original problem.

4.2. Pricing problem

The resulting subproblem of the reformulation of the positional model is a single machine scheduling problem associated with each machine, $i \in I$. This problem is classified as $1 \mid s_{lj} \mid \Sigma T_j$, and is NP-hard (Du & Leung, 1990). In this problem, consideration of the sequence-dependent setup times increases the complexity (Pinedo, 2008). Furthermore, this problem is even more difficult when the total tardiness is the performance criterion, and it thus has been relatively less studied (Allahverdi & Soroush, 2008). Therefore, it is possible that no published study has employed CG to solve the given problem.

In the pricing problem, the objective function is the equation (36), and the constraints are the same as those of the positional model when only one machine (PM_{PP}) is considered. The pricing problem is to find a job sequence or column s for each machine i to minimize the reduced cost r_{is}.

4.3. Heuristics used in the column generation

Three heuristics are proposed. The first is to generate a set of quality feasible initial columns. The second is to solve faster pricing problems, but with no guarantee of optimality. The third can either generate initial columns or solve the subproblem. Before describing the heuristics, it is necessary to define how the cost of each solution is calculated. From the heuristic solution, each job is inserted in the machine that obtains the lowest tardiness.

4.3.1. Iterated local search for the initial solution

The iterated local search (ILS) involves iteratively perturbing a local optimal solution, then applying a local search procedure to obtain a new local optimal solution, and, finally, using an acceptance criterion for deciding from which of these solutions to continue the search (Lourenço, Martin & Stützle, 2002). This metaheuristic was chosen because of its features, including simplicity, robustness, effectiveness, easy implementation, and successful implementations for different optimization problems, such as the traveling salesman, job shop, and flow shop (Lourenço, Martin & Stützle, 2002).

The ILS algorithm is comprised of four components. They are a procedure, *GenerateInitialSolution*, which generates an initial solution s_0; a procedure, *Perturbation*, which modifies the current solution s leading to some intermediate solution s'; a procedure, *LocalSearch*, which returns an improved solution s''; and an *AcceptanceCriterion*, which decides to which solution the next perturbation is applied. The steps of the ILS algorithm are outlined as follows.

***GenerateInitialSolution*:** This procedure consists of generating *ns* random job sequences and choosing the best job sequence to be the initial solution, s_0.

***Perturbation*:** Various types of perturbations were tested. The one that presents the best performance is a randomly selected job from a job sequence that changes its position with that of the next job. If the select job is in the last position, it is exchanged with the previous job.

***LocalSearch*:** This procedure uses the one-*opt* neighborhood in which each move changes a job with the next job. For searching, the steepest descent method is used.

***AcceptanceCriterion*:** This procedure uses one of the following rules: accept every new solution since it is up to *bp*% of the best solution, or accept the best solution obtained so far in the search. This corresponds to a strategy for the diversification and intensification of the search, respectively. The job sequence allocated to each machine in the final solution ILS algorithm may not be optimal. Therefore is used a fix-and-optimize heuristic as following: fix the jobs on each machine according to the final solution ILS algorithm, and execute the positional model.

4.3.2. Constructive heuristic for pricing problem

The constructive heuristic (CH) consists of iteratively building a partial job sequence by assessing—through a local search of jobs that have not yet been chosen—the one that is better to be inserted for finding the lowest reduced cost. This heuristic is terminated when the evaluation of all remaining jobs does not find a partial job sequence with a negative reduced cost. The steps of this heuristic are outlined below.

Step 1: Calculate the reduced cost of all possible pairs of jobs. If the pair of jobs with a lower reduced cost is negative, then this pair of jobs is the initial partial job sequence of Step 2. Otherwise, the heuristic is closed.

Step 2: Choose the next job to be inserted in the partial job sequence. For all jobs that do not belong to the partial job sequence, insert it into the first position of the partial job sequence, perform a local search, and calculate the reduced cost. If it is negative, insert it in the master problem. Thus, multiple columns are inserted per iteration. Among the found partial job sequences, choose the one with the lowest reduced cost to be the new partial job sequence.

Step 3: Repeat Step 2 to find at least a partial job sequence with a negative reduced cost.

4.3.3. Linear relaxation-based heuristic for an initial solution or pricing problem

The linear relaxation-based heuristic (LRBH) involves solving the linear relaxation of some mixed integer linear programming model. It uses the information of the fractional solution to fix some variables, and then runs the model again with active integrity constraints. This heuristic has been successfully applied to many different problems, such as the respective project scheduling (Azizoglu, Çetinkaya, & Pamir, 2015), logistic network design (Thanh, Péton, & Bostel, 2010), general assignment (French & Wilson, 2007), and lot-sizing (Hardin, Nemhauser, & Savelsbergh, 2007) problems.

We propose two heuristic algorithms based on linear relaxation of the positional model. One is used to generate the initial columns; the other is used to solve the pricing problem. Therefore, the first considers the entire problem; the second considers the single machine scheduling problem. The jobs that do not appear with a fractional value on a machine i in the solution of the relaxed problem of the positional model will not be considered in the model solution with active integrity constraints. The main advantage of this method is to yield feasible solutions of good quality within a limited computational time. The steps of this heuristics are outlined below.

Step 1: Perform relaxed positional model. Job j, for which variables x_{ijp} obtained a zero value in all positions p of machine i, is considered unavailable for this machine in the next step. This is achieved by setting all variables, x_{ijp}, to zero for machine i.

Step 2: The positional model is run with the jobs fixed according to the condition and active integrality constraints.

4.4. Hybrid column generation algorithm

The proposed CG algorithm is called the hybrid column generation (HCG) algorithm because it uses heuristics to generate the initial columns and solve the pricing problem. The overall steps of this algorithm are outlined as follows.

Step 1 (Dantzig–Wolfe decomposition): Reformulate the original problem as a set partitioning problem via Dantzig–Wolfe decomposition, which is explained in Section 4.1.

Step 2 (Construction of a feasible initial solution): Generate the initial columns by one of the two heuristics explained in Sections 4.3.1 and 4.3.3.

Step 3 (Restricted master problem): Solve $\mathrm{RMP}_{\mathrm{LR}}$ to find the dual variable values.

Step 4 (Pricing problem): At this stage, employ three different algorithms in a specific order. The first two are heuristics, and the third is an exact method. They are applied in the following order: CH, $\mathrm{LRBH}_{\mathrm{PP}}$, and $\mathrm{PM}_{\mathrm{PP}}$ (proposed model considering a single machine). If one of the algorithms finds a solution with a negative reduced cost, then the associated column is added to $\mathrm{RMP}_{\mathrm{LR}}$ and Step 3 is repeated; otherwise, proceed to the next algorithm. All algorithms insert multiple columns with negative reduced costs per iteration. If the third algorithm does not find columns with negative reduced costs, proceed to Step 5.

Step 5 (Evaluation of optimality): The current solution of $\mathrm{RMP}_{\mathrm{LR}}$ is a lower bound (*lb*) for the original problem. Then, RMP is solved with active integrality constraints to find an upper bound (*ub*) for the original problem. If *lb* is equal to *ub,* then the solution is optimal; otherwise, the relative optimality gap is calculated as 100 * (*ub* – *lb*) / *lb*.

5. Computational experiments

To evaluate the methods, instances generated based on Lopes and Carvalho (2007) and Rocha, Ravetti, Mateus, and Pardalos (2008) were used as follows. Small-size instances were generated using the number of jobs, $n \in \{10, 15, 20\}$, and number of machines, $m \in \{2, 3, 4\}$. For medium-size instances, we used $n \in \{20, 30, 40, 50\}$ and $m \in \{2, 4, 6, 8, 10\}$, as in Lopes and de Carvalho (2007), and the processing times $p_{jl} = U[10, 80]$ and setup times $s_{ijl} = U[20, 40]$. Since the setup times were generated randomly, they needed to be corrected to satisfy the triangular inequality. The triangular inequality states that, for any three jobs *j*, *l*, *k* requiring the same resource (machine *i*), the inequality $s_{ijk} \leq s_{ijl} + p_{il} + s_{ilk}$ is ensured. In order to do that the same procedure from Rocha, Ravetti, Mateus, and Pardalos (2008) was used. The due dates were generated following the method of Rocha, Ravetti, Mateus, and Pardalos (2008), $d_j = U$[maximal processing time, $2h/q$]. Parameter *h* is the makespan of identity solution (1, 2, …, *n*), where each job is assigned to a machine capable of finishing it first. Parameter *q* indicates the congestion level of the production system. The larger the *q*, the more congested the system will be, and the more tardy the jobs will be. Both authors used *q* varying from one to five. For each combination of *n*, *m* and *q* are ten instances randomly generated using different seeds. Then, the tests consist of 300 and 450 instances of small and medium sizes, respectively.

The mathematical models and the HCG algorithm were implemented with the C++ API for Concert Technology and were solved with IBM ILOG CPLEX 12.5. Tests were performed on a Dell Inspiron notebook, Intel Core i5-2430M 2.40-GHz processor with 4 GB of memory and the Windows 7 operating system. The maximum time allowed for running any algorithm was 3,600 s. If the solver was unable to find the optimal solution, the best integer solution found was reported.

From the initial tests, it was verified that the system congestion level influenced the effectiveness of heuristics in finding good solutions for initial columns. The heuristic based on the proposed model did not find feasible solutions of quality when the production system was slightly congested. Therefore, for $q = 1$ and 2, the ILS algorithm was used; and for $q = 3$, 4 and 5, the LRBH_{IS} approach was used. For the ILS algorithm, its parameters and values were the number of job sequences, *ns*, randomly generated (1,000), number of iterations (up to 1,000) or (200, when a better solution than all current ones was not obtained), and relative percentage deviation from the best solution, *bp*% (10%).

The computational time limit employed by the HCG heuristic procedures was 1,800 s. The computational time to solve the pricing problem based on the positional model (PM_{PP}) was limited to 3,600 s less than the time consumed by the previous heuristics.

The test results comprised two groups. The first compared Wagner (W), Manne (M), and positional (P) models using small-size instances. The second compare the positional model and HCG method using medium-size instances. Tables 1 to 5 show the test results for the first group; Tables 6 to 10 show the test results for the second group. The meaning of the table headings is the following: *n* denotes the number of jobs, *m* is the number of machines, and *q* represents the production system congestion level. In Tables 3 to 5 and 8 to 10, for each combination of *n* and *m,* there are three lines for the minimum, mean, and maximum, respectively.

Tables 1 and 6 show the average percentage deviation between the best feasible solution (*bfs*) and the linear relaxation (*lr*) obtained by the method, which is calculated as $100 * (bfs - lr) / bfs$. Tables 2 and 7 show the number of instances that were unsolved in terms of optimality for each group. Tables 3 and 8 show the average percentage gap, which is calculated as $100 * (bfs – blb) / bfs$, where *blb* is the lower bound obtained by the method. Tables 4 and 9 show the relative percentage deviation, which is calculated as $100*(msi – bsi) / bsi$, where *msi* is the instance solution value using one of the methods, and *bsi* is the best known objective function value for the instance. Tables 5 and 10 show the elapsed CPU times in seconds to solve the instances.

Table 11 shows the performance of the components of the proposed HCG method. For each combination of jobs and machines, the five lines correspond to five congestion levels. The column sets, IS, CH, LRBH_{PP} and PM_{PP}, represent the average of the results for the initial solution, column generation by constructive heuristic, column generation by restricted positional model, and column generation by positional model, respectively. The average relative deviation of the initial solution (IS) in relation to the final solution (FS), the average number of iterations, the average generated columns, and the average runtime are shown in columns $\Delta_{\text{IS-FS}}$, Iter, Col, and Time, respectively. $\Delta_{\text{IS-FS}}$ is calculated as $100*(\text{FS} – \text{IS}) / \text{IS}$.

The test results from the first group showed that, when the system congestion level (parameter q) increased, the Wagner and Manne models had greater difficulty solving the instances, while the positional model experienced less difficulty (see Tables 1 to 5). Analyzing the instances of the congestion levels $q = 1$ and $q = 2$, it was observed that there are jobs with so loose due dates which they would not be tardy, even if they were processed last. Tests were conducted to prove this hypothesis. These jobs were removed and the positional model was run. The identified solution consumed less computational time and achieved a 0% gap. If the extracted jobs were inserted at the end of the job sequences, these jobs would not be tardy. This fact deserves further study in future work.

The positional model did not obtain linear relaxations greater than zero in any of the tested instances for $q = 1$ because the due dates were very loose. At congestion levels $q = 1$ and $q = 2$, 32 and 7 optimal solutions equal to zero were respectively obtained. At other congestion levels, no optimal solution was equal to zero. We emphasize that optimal solutions equal to zero were not considered in the calculations shown in Table 1 because they did not help assess the ability of the methods to find tight linear relaxations. In comparing the Wagner and positional models, the former did not obtain linear relaxations greater than zero in any instance tested. The latter obtained linear relaxations greater than zero starting at the congestion level $q = 2$.

The positional model did not solve the problem until the optimality in nine instances. All these instances were of the congestion level $q = 2$, which had the worst linear relaxations. The congestion level $q = 1$ only obtained more optimal solutions than the congestion level $q = 2$ because of the large number of optimal solutions equal to zero. The Wagner and Manne models did not solve the problem until the optimality of 202 and 152 instances, respectively.

The positional model presented a gap greater than 0%, precisely 9.29% on average, only at the congestion level $q = 2$. The only level at which it did not find the optimal solution of all instances was even smaller than that of the Manne model, which obtained 16.26% on average (see Table 3). The positional model did not obtain a better final solution than the Manne model in only two instances of congestion level $q = 2$ (see Table 4). The Wagner model obtained worse final solutions than the Manne and positional models, especially in the congestion level $q = 2$, with an average of 42.09% worse than the best solutions found.

The models had the same general trend when considering the runtime to solve the instances. The trend only changed at congestion level $q = 1$ because of the large number of optimal solutions equal to zero. For congestion level $q = 1$, the Manne model obtained a smaller runtime than the positional model at 1.15 s versus 70.67 s on average. At congestion level $q = 3$, the positional model was up to two orders of magnitude faster than the other models. For $q = 4$ and $q = 5$, it was up to three orders of magnitude faster (see Table 5).

Of all the analyzed criteria, the Wagner model obtained performance well below that of the positional model and even in relation to the Manne model. This aspect would be investigated if there were other studies with similar results. Thus, the work of Lange and Werner (2015) was found to use models based on precedence and position variables on a parallel machine approach to minimize the total tardiness of a single-track train scheduling problem. In tests performed with instances of ten jobs or more, the model based on position variables was not able to obtain an optimal solution within 2 h of runtime, whereas the model based on precedence variables obtained the optimal solution in 2.02 s on average.

Table 1 Linear relaxation deviation (%) of instances solved by Wagner (W), Manne (M), and positional (P) models for small-size instances.

n	*m*	*q* = 1			*q* = 2			*q* = 3			*q* = 4			*q* = 5		
		W	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**
10	2	100.00	47.59	100.00	100.00	87.08	74.03	100.00	92.23	13.93	100.00	92.63	5.71	100.00	92.07	3.04
10	3	100.00	35.73	100.00	100.00	77.22	62.01	100.00	87.70	10.90	100.00	88.04	6.40	100.00	86.95	3.50
15	2	100.00	50.00	100.00	100.00	82.11	96.01	100.00	97.40	24.42	100.00	97.96	9.25	100.00	97.83	5.67
15	3	100.00	7.41	100.00	100.00	96.68	83.76	100.00	98.00	19.23	100.00	97.46	7.00	100.00	96.85	3.69
20	3	100.00	18.33	100.00	100.00	72.25	93.79	100.00	97.66	25.40	100.00	98.08	10.47	100.00	98.00	5.62
20	4	100.00	31.90	100.00	100.00	80.59	94.66	100.00	94.43	23.66	100.00	95.52	9.20	100.00	95.55	4.89
Average		100.00	31.83	100.00	100.00	82.66	84.04	100.00	94.57	19.59	100.00	94.95	8.00	100.00	94.54	4.40

Table 2 Unsolved number of instances using Wagner, Manne, and positional models for small-size instances.

n	*m*	*q* = 1			*q* = 2			*q* = 3			*q* = 4			*q* = 5		
		W	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**
10	2	0	0	0	1	0	0	2	0	0	2	3	0	8	6	0
10	3	0	0	0	0	0	0	7	1	0	8	3	0	8	5	0
15	2	0	0	0	9	2	0	10	10	0	10	10	0	10	10	0
15	3	1	0	0	7	2	0	10	10	0	10	10	0	10	10	0
20	3	4	0	0	10	4	4	10	10	0	10	10	0	10	10	0
20	4	5	0	0	10	6	5	10	10	0	10	10	0	10	10	0
Total		10	0	0	37	14	9	49	41	0	50	46	0	56	51	0

Table 3 Gap (%) of instances solved by Wagner, Manne, and positional models for small-size instances.

n	*m*	*q* = 1			*q* = 2			*q* = 3			*q* = 4			*q* = 5		
		W	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**
10	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	3.41	0	0	8.33	0	0	6.75	7.61	0	31.37	19.03	0
		0	0	0	34.12	0	0	69.81	0	0	44.51	26.36	0	59.58	43.44	0
10	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	21.39	2.19	0	36.15	7.03	0	26.67	10.07	0
		0	0	0	0	0	0	46.15	21.94	0	63.78	28.70	0	53.98	31.61	0
15	2	0	0	0	0	0	0	94.21	57.58	0	96.23	80.69	0	96.66	81.49	0
		0	0	0	78.69	19.21	0	97.88	79.60	0	98.16	87.36	0	98.05	86.71	0
		0	0	0	100.00	100.00	0	100.00	97.76	0	100.00	94.11	0	99.46	94.16	0

15	3	0	0	0	0	0	0	94.79	52.42	0	92.79	73.97	0	92.50	67.29	0
		0.74	0	0	69.16	14.72	0	98.77	78.47	0	98.08	86.17	0	97.43	83.15	0
		7.41	0	0	100.00	100.00	0	100.00	99.47	0	100.00	96.72	0	100.00	93.26	0
20	3	0	0	0	96.13	0	0	97.08	82.37	0	97.88	85.42	0	96.91	86.16	0
		35.50	0	0	99.15	31.59	24.52	99.29	93.07	0	99.39	93.90	0	99.02	93.54	0
		100.00	0	0	100.00	100.00	73.37	100.00	100.00	0	100.00	100.00	0	100.00	99.27	0
20	4	0	0	0	94.29	0	0	95.74	76.19	0	93.35	82.65	0	92.77	82.64	0
		45.68	0	0	97.58	32.07	31.20	98.92	86.72	0	97.27	88.95	0	98.23	88.45	0
		100.00	0	0	100.00	89.15	82.46	100.00	92.32	0	100.00	93.34	0	100.00	92.48	0
Average		13.65	0.00	0.00	58.00	16.26	9.29	70.77	56.68	0.00	72.63	61.84	0.00	75.13	63.49	0.00

The above results indicate that the positional model had the expected effect and that, unlike the time-indexed formulations, the positional model could be solved in a computational time (see Table 5) as reasonable as those of the models based on the formulation of type 1 (see Section 1). The positional model uses positional variables, similar to the Wagner model, and obtains the best computational performance compared to it and the Manne model. It is thus numerically proved that the proposed innovations are the contributing factors of the achieved improvement.

The results of the positional model and HCG method, when used to solve medium-size instances, are shown in Tables 6 to 11. The positional model presents the same trend observed during the tests with the small-size instances. That is, for congestion levels $q = 1$ and $q = 2$, it has more difficulty in obtaining optimal solutions. That it did not obtain a linear relaxation greater than zero for any instance (see Table 6) is proof.

The positional model found 38 optimal solutions for the congestion level $q = 1$. All of these solutions were equal to zero because it found more optimal solutions than for congestion level $q = 2$, where it found only seven optimal solutions, of which four equaled zero. Meanwhile, the HCG method found the optimal solution of 89 and 28 of the 90 tested instances of congestion levels $q = 1$ and $q = 2$, respectively. This because of the good initial solutions found by the ILS method, as shown in the $\Delta_{IS\text{-}FS}$ column of Table 11 for the congestion levels $q = 1$ and $q = 2$. The average improvement between the final and initial solutions was 0.15% and 5.81%, respectively. The lower is this improvement, the closer are the initial and final solutions.

The above results demonstrate the quality of the initial solution because the final solution also had quality. The positional model had difficulty solving instances with loose due dates using the ILS method to generate the initial solutions, which proved advantageous. The quality of the initial solutions was also verified when the LBRH$_{IS}$ method was used. It was observed that the average improvement of the final solution in relation to the initial solution was 3.43%, 0.69%, and 0.20% for the congestion levels $q = 3$, $q = 4$, and $q = 5$, respectively.

Table 4 Relative deviation (%) of instances solved by Wagner, Manne, and positional models for small-size instances.

n	*m*	*q* = 1			*q* = 2			*q* = 3			*q* = 4			*q* = 5		
		W	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**
10	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0.30	0	0	0	0	0	0.11	0	0
		0	0	0	0	0	0	2.97	0	0	0	0	0	1.07	0	0
10	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0.37	0	0	0.14	0	0
		0	0	0	0	0	0	0	0	0	1.90	0	0	1.35	0	0
15	2	0	0	0	0	0	0	9.76	0	0	7.19	0	0	9.48	0	0
		0	0	0	35.19	0	0	20.51	0.30	0	13.10	0.61	0	13.08	1.31	0
		0	0	0	73.33	0	0	43.51	1.19	0	18.62	4.32	0	19.89	3.41	0
15	3	0	0	0	0	0	0	12.90	0	0	4.78	0	0	7.24	0	0
		0	0	0	48.28	0.41	0	29.20	0.61	0	19.12	0.23	0	16.79	1.20	0
		0	0	0	100.00	4.12	0	51.48	4.18	0	34.71	2.19	0	24.61	3.59	0
20	3	0	0	0	59.05	0	0	28.80	0	0	20.40	0	0	15.93	0.34	0
		23.23	0	0	89.23	1.26	0.70	50.41	8.15	0	35.66	2.20	0	26.55	3.03	0
		100.00	0	0	100.00	6.17	6.98	59.52	29.02	0	52.26	7.32	0	39.40	6.53	0
20	4	0	0	0	59.34	0	0	28.27	0	0	27.25	0	0	18.08	0	0
		29.84	0	0	79.84	4.73	0.62	48.64	4.04	0	33.67	3.03	0	28.00	3.01	0
		100.00	0	0	97.38	19.70	6.20	63.22	12.63	0	42.75	8.86	0	41.16	6.60	0
Average		8.85	0.00	0.00	42.09	1.07	0.22	24.84	2.18	0.00	16.99	1.01	0.00	14.11	1.42	0.00

Table 5 Runtimes of instances solved by Wagner, Manne, and positional models for small-size instances.

n	*m*	*q* = 1			*q* = 2			*q* = 3			*q* = 4			*q* = 5		
		W	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**	**W**	**M**	**P**
10	2	0.31	0.03	0.23	102.10	0.45	0.56	908.75	35.32	0.27	978.00	491.43	0.23	2515.91	938.94	0.17
		53.26	0.28	7.21	812.14	7.61	3.34	2081.52	621.68	0.54	2808.32	1775.67	0.40	3430.80	2768.92	0.32
		367.76	1.26	49.89	3600.00	44.71	23.12	3600.00	2707.34	1.09	3600.00	3600.00	0.67	3600.00	3600.00	0.55
10	3	1.67	0.08	0.20	139.31	1.08	0.23	1746.34	130.98	0.23	1852.81	365.76	0.09	2741.90	707.84	0.11
		20.61	0.36	6.59	569.99	14.68	0.95	3307.03	803.59	0.44	3405.38	1774.03	0.32	3438.27	2632.78	0.21
		111.40	1.87	30.11	1751.61	63.80	3.14	3600.00	3600.00	0.86	3600.00	3600.00	0.64	3600.00	3600.00	0.37
15	2	15.77	0.16	0.34	599.47	2.96	5.55	3600.00	3600.00	1.84	3600.00	3600.00	0.89	3600.00	3600.00	0.80
		502.65	0.45	15.05	3299.95	836.75	99.69	3600.00	3600.00	7.81	3600.00	3600.00	4.11	3600.00	3600.00	1.88
		2947.96	1.05	102.46	3600.00	3600.00	287.26	3600.00	3600.00	13.07	3600.00	3600.00	9.13	3600.00	3600.00	5.01
15	3	2.90	0.20	0.41	1279.44	3.68	1.95	3600.00	3600.00	0.67	3600.00	3600.00	0.53	3600.00	3600.00	0.44
		460.20	0.90	4.63	3220.25	1094.91	109.03	3600.00	3600.00	4.13	3600.00	3600.00	1.37	3600.00	3600.00	0.66

		3600.00	5.38	27.72	3600.00	3600.00	536.92	3600.00	3600.00	11.92	3600.00	3600.00	5.96	3600.00	3600.00	1.11
20	3	92.62	0.39	1.00	3600.00	2.90	3.56	3600.00	3600.00	3.90	3600.00	3600.00	3.07	3600.00	3600.00	2.06
		2117.63	1.94	81.08	3600.00	1585.96	1911.78	3600.00	3600.00	42.22	3600.00	3600.00	13.92	3600.00	3600.00	6.88
		3600.00	6.33	438.75	3600.00	3600.00	3600.00	3600.00	3600.00	101.15	3600.00	3600.00	23.26	3600.00	3600.00	15.52
20	4	334.73	0.62	1.53	3600.00	33.93	53.51	3600.00	3600.00	2.81	3600.00	3600.00	1.76	3600.00	3600.00	1.34
		2067.25	2.95	309.45	3600.00	2564.47	2073.55	3600.00	3600.00	26.84	3600.00	3600.00	11.44	3600.00	3600.00	3.74
		3600.00	7.66	2602.25	3600.00	3600.00	3600.00	3600.00	3600.00	50.09	3600.00	3600.00	28.74	3600.00	3600.00	9.02
Average		870.26	1.15	70.67	2517.05	1017.40	699.72	3298.09	2637.54	13.67	3435.62	2991.62	5.26	3544.84	3300.28	2.28

Table 6 Linear relaxation deviation (%) of instances solved by the positional model for medium-size instances.

n	*m*	*q* = 1	*q* = 2	*q* = 3	*q* = 4	*q* = 5
20	2	100.00	100.00	30.06	11.81	6.10
30	2	100.00	100.00	31.03	10.65	5.75
30	4	100.00	100.00	31.62	11.32	5.88
40	4	100.00	100.00	44.29	14.12	7.19
40	6	100.00	100.00	38.19	12.12	6.27
40	8	100.00	100.00	34.15	10.70	5.18
50	6	100.00	100.00	33.83	11.79	5.76
50	8	100.00	100.00	38.91	12.84	6.07
50	10	100.00	100.00	35.66	11.18	5.74
Average		100.00	100.00	35.31	11.84	5.99

Table 7 Unsolved number of instances using the positional model and HCG for medium-size instances.

n	*m*	*q* = 1		*q* = 2		*q* = 3		*q* = 4		*q* = 5	
		P	**HCG**	**P**	**HCG**	**P**	**HCG**	**P**	**HCG**	**P**	**HCG**
20	2	2	1	5	6	0	10	0	10	0	10
30	2	3	0	9	5	10	10	6	10	0	10
30	4	2	0	9	8	6	10	2	9	0	10
40	4	5	0	10	6	10	10	10	10	6	10
40	6	6	0	10	7	10	10	7	10	0	10
40	8	9	0	10	8	10	10	2	10	0	9
50	6	8	0	10	8	10	10	10	10	8	10
50	8	7	0	10	6	10	10	10	10	6	10
50	10	10	0	10	8	10	10	8	10	2	10
Total		52	1	83	62	76	90	55	89	22	89

The initial solution being close to the final solution did not mean that the HCG method had quality. Rather, the average gap of the final solutions, 0.03%, 12.68%, 2.85%, 0.58% and 0.17% for congestion levels q = 1 to q = 5, respectively, demonstrated this point. It also showed that the lower bound obtained by the HCG method was tightened (see Table 8). In addition, the HCG method found the best final solutions compared to the positional model, especially in congestion levels q = 1 and q = 2, with average deviation in relation to the best solution equal to 0% and 0.05% versus 23.05% and 52.15% of the positional model, respectively (see Table 9). At congestion levels q = 3 to q = 5, the difference between the methods was smaller; however, the HCG method still presented better final solutions.

Meanwhile, at congestion levels q = 1 and q = 2, the instances in which the ILS method found the initial solution equal to zero did not have to proceed through the solution methods of the subproblems. This is because the runtimes of these procedures were zero. The average total runtime of the HCG method (Table 10) is smaller than the sum of the average runtimes of the components of the HCG method (Table 11). In terms of the average values presented in Table 11, only those instances that had to proceed through the subproblem-solving procedures were considered.

At all congestion levels, the HCG method ran faster than the positional model (see Table 10). The largest difference was observed at congestion level q = 1: 99.30 s versus 2,306.96 s. This is because of the quality of the initial solutions generated by the ILS and LBRH$_{IS}$ methods, and by the heuristic methods to solve the subproblems. The CH method used fewer runtimes; however, it performed more iterations and generated more columns because dual variables remained far from the optimal values. The LRBH$_{PP}$ method used a longer runtime, performed fewer iterations, and generated fewer columns. The use of these two heuristics was advantageous when the exact PM$_{PP}$ procedure was used. The dual variables were closer to the optimal value and fewer iterations were needed to find their optimal value (see Table 11).

Table 8 Gap of instances solved by the positional model and HCG for medium-size instances.

n	*m*	*q* = 1		*q* = 2		*q* = 3		*q* = 4		*q* = 5	
		P	**HCG**	**P**	**HCG**	**P**	**HCG**	**P**	**HCG**	**P**	**HCG**
20	2	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.05	0.00	0.04
		17.64	0.26	37.06	1.45	0.00	0.30	0.00	0.15	0.00	0.12
		96.58	2.61	100.00	9.00	0.00	0.49	0.00	0.45	0.00	0.57
30	2	0.00	0.00	0.00	0.00	5.20	6.42	0.00	0.04	0.00	0.02
		30.00	0.00	89.55	9.94	21.86	13.12	2.85	2.25	0.00	0.12
		100.00	0.00	100.00	54.59	44.27	18.50	11.02	4.34	0.00	0.79
30	4	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.04
		20.00	0.00	88.24	13.01	12.93	1.02	0.39	0.27	0.00	0.14
		100.00	0.00	100.00	42.31	39.88	4.03	2.88	0.44	0.00	0.56
40	4	0.00	0.00	80.56	0.00	14.15	0.06	2.91	0.03	0.00	0.03
		50.00	0.00	98.06	13.01	35.94	4.66	7.90	0.69	1.82	0.22
		100.00	0.00	100.00	51.50	57.87	16.27	14.01	4.90	4.11	0.87
40	6	0.00	0.00	91.02	0.00	3.56	0.08	0.00	0.06	0.00	0.04

		53.33	0.00	99.10	13.63	24.60	1.39	3.17	0.30	0.00	0.19
		100.00	0.00	100.00	72.34	38.04	4.27	6.76	0.88	0.00	0.65
40	8	0.00	0.00	100.00	0.00	2.98	0.11	0.00	0.08	0.00	0.00
		90.00	0.00	100.00	6.71	18.71	1.35	1.06	0.32	0.00	0.12
		100.00	0.00	100.00	18.75	41.12	3.50	7.60	0.66	0.00	0.46
50	6	0.00	0.00	100.00	0.00	16.92	0.04	1.97	0.03	0.00	0.02
		80.00	0.00	100.00	14.34	29.25	0.80	7.97	0.17	1.97	0.19
		100.00	0.00	100.00	32.69	52.94	4.67	12.05	0.57	5.56	1.12
50	8	0.00	0.00	100.00	0.00	21.93	0.21	2.32	0.05	0.00	0.04
		70.00	0.00	100.00	29.76	32.33	1.42	7.58	0.72	1.19	0.28
		100.00	0.00	100.00	100.00	59.81	3.56	13.03	1.70	3.54	0.68
50	10	100.00	0.00	100.00	0.00	13.10	0.14	0.00	0.03	0.00	0.09
		100.00	0.00	100.00	12.24	26.41	1.58	4.01	0.37	0.37	0.18
		100.00	0.00	100.00	25.32	59.08	5.81	12.22	1.11	2.08	0.48
Average		56.77	0.03	90.22	12.68	22.45	2.85	3.88	0.58	0.59	0.17

Table 9 Relative deviation of instances solved by the positional model and HCG for medium-size instances.

n	*m*	*q* = 1		*q* = 2		*q* = 3		*q* = 4		*q* = 5	
		P	**HCG**	**P**	**HCG**	**P**	**HCG**	**P**	**HCG**	**P**	**HCG**
20	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.00	0.00	3.83	0.00	0.02	0.00	0.00	0.01	0.01	0.00
		0.00	0.00	27.50	0.00	0.12	0.00	0.00	0.06	0.10	0.00
30	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		10.00	0.00	43.76	0.00	1.77	1.02	0.03	0.42	0.00	0.00
		100.00	0.00	96.49	0.00	6.89	6.46	0.24	3.34	0.02	0.00
30	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		5.63	0.00	32.85	0.00	1.46	0.00	0.00	0.02	0.00	0.00
		56.25	0.00	66.67	0.00	6.03	0.00	0.00	0.08	0.04	0.00
40	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		17.80	0.00	61.28	0.42	4.30	0.42	0.53	0.15	0.13	0.04
		94.49	0.00	100.00	4.20	9.28	3.81	2.15	1.46	0.80	0.35
40	6	0.00	0.00	6.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		26.60	0.00	62.08	0.00	4.15	0.09	0.24	0.04	0.00	0.06
		100.00	0.00	90.87	0.00	7.39	0.92	1.18	0.37	0.04	0.51
40	8	0.00	0.00	26.54	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		16.21	0.00	56.69	0.00	2.83	0.00	0.09	0.04	0.05	0.00
		88.74	0.00	87.88	0.00	6.83	0.00	0.38	0.37	0.22	0.00

50	6	0.00	0.00	5.93	0.00	2.69	0.00	0.00	0.00	0.00	0.00
		57.93	0.00	54.55	0.00	8.18	0.00	1.68	0.01	0.31	0.09
		100.00	0.00	93.41	0.00	18.31	0.00	3.96	0.10	2.56	0.91
50	8	0.00	0.00	51.19	0.00	1.60	0.00	0.08	0.00	0.00	0.00
		25.83	0.00	78.85	0.00	9.35	0.00	1.65	0.00	0.02	0.03
		100.00	0.00	99.69	0.00	17.75	0.00	3.36	0.00	0.19	0.14
50	10	0.00	0.00	48.56	0.00	1.36	0.00	0.00	0.00	0.00	0.00
		47.48	0.00	75.44	0.00	6.25	0.00	0.54	0.02	0.13	0.00
		95.53	0.00	93.33	0.00	16.53	0.00	2.74	0.24	0.31	0.00
Average		23.05	0.00	52.15	0.05	4.26	0.17	0.53	0.08	0.07	0.02

Table 10 Runtime of instances solved by the positional model and HCG for medium-size instances.

n	*m*	q = 1		q = 2		q = 3		q = 4		q = 5	
		P	**HCG**	**P**	**HCG**	**P**	**HCG**	**P**	**HCG**	**P**	**HCG**
20	2	0.94	0.47	6.83	7.92	25.65	231.6	7.47	88.11	2.89	68.89
		738.03	376.77	2134.70	602.73	342.37	719.31	31.33	226.88	14.23	129.91
		3600.00	3600.00	3600.00	2741.85	1044.65	1822.68	65.26	330.30	22.00	194.89
30	2	15.76	3.32	3340.98	176.50	3600.00	3167.13	188.03	2696.51	81.39	1737.33
		1125.44	85.16	3574.10	1740.74	3600.00	3520.01	2526.55	3154.82	382.69	2687.22
		3600.00	242.83	3600.00	3600.00	3600.00	3600.00	3600.00	3600.00	1131.74	3129.83
30	4	67.61	0.98	2755.77	44.21	715.84	254.70	65.91	108.58	26.94	75.25
		1098.96	8.06	3515.58	1918.86	2731.45	632.59	1384.32	238.65	70.43	124.59
		3600.00	13.63	3600.00	3600.00	3600.00	942.26	3600.00	484.40	127.70	177.29
40	4	387.96	1.87	3600.00	135.74	3600.00	2330.21	3600.00	1255.75	438.96	690.55
		2055.61	31.16	3600.00	1987.75	3600.00	3138.02	3600.00	1644.49	2448.41	1005.55
		3600.00	150.34	3600.00	3600.00	3600.00	3600.00	3600.00	2017.85	3600.00	1492.02
40	6	239.95	4.42	3600.00	141.57	3600.00	511.79	244.00	367.33	94.24	145.31
		2637.08	28.75	3600.00	2021.51	3600.00	813.54	2957.90	475.75	366.02	228.08
		3600.00	71.76	3600.00	3600.00	3600.00	1218.78	3600.00	586.44	1271.72	384.71
40	8	196.09	34.74	3600.00	162.01	3600.00	386.18	150.63	127.80	43.84	70.93
		3259.61	44.15	3600.00	1202.32	3600.00	429.69	1052.81	235.09	97.66	100.34
		3600.00	67.73	3600.00	3600.00	3600.00	504.38	3600.00	447.22	197.25	122.54
50	6	972.85	12.00	3600.00	260.71	3600.00	1533.36	3600.00	892.98	429.99	556.81
		3325.89	80.24	3600.00	2698.00	3600.00	2140.53	3600.00	1239.13	3008.76	770.80
		3600.00	282.61	3600.00	3600.00	3600.00	3215.01	3600.00	1432.29	3600.00	921.35
50	8	215.78	9.11	3600.00	342.90	3600.00	714.59	3600.00	549.28	489.19	334.75
		2922.06	96.96	3600.00	2221.26	3600.00	870.90	3600.00	642.89	2615.01	498.12

		3600.00	191.26	3600.00	3600.00	3600.00	1437.68	3600.00	749.11	3600.00	645.51
50	10	3600.00	69.51	3600.00	367.12	3600.00	521.35	890.94	279.65	314.01	183.63
		3600.00	142.44	3600.00	2076.51	3600.00	560.36	3241.00	463.30	1208.79	278.33
		3600.00	223.52	3600.00	3600.00	3600.00	627.92	3600.00	558.22	3600.00	535.46
Average		2306.96	99.30	3424.93	1829.96	3141.54	1425.00	2443.77	924.55	1134.67	646.99

Table 11 Performances of the proposed HCG components.

n	*m*	*q*	**IS**		**CH**			**LRBH$_{PP}$**			**PM$_{PP}$**		
			Δ_{IS-FS}	**Time**	**Iter**	**Col**	**Time**	**Iter**	**Col**	**Time**	**Iter**	**Col**	**Time**
20	2	1	0.00	75.96	5.50	34.65	0.37	1.17	1.00	0.79	1.50	2.17	548.99
		2	0.00	106.50	11.78	123.67	1.26	3.67	10.56	31.39	6.44	38.11	740.51
		3	1.35	118.60	21.50	135.45	1.47	32.20	191.50	277.05	12.90	33.60	321.42
		4	0.02	16.45	24.70	155.61	1.95	28.60	132.10	128.54	9.00	15.60	79.25
		5	0.01	9.73	23.80	249.90	1.67	27.30	110.70	72.70	9.40	19.10	45.12
30	2	1	0.00	147.10	2.50	26.25	0.82	1.00	0.00	0.15	1.00	0.00	0.24
		2	3.58	315.47	20.78	130.90	13.08	1.56	4.22	658.61	1.67	5.33	1141.31
		3	1.21	304.37	42.80	269.64	16.26	6.80	104.30	1578.84	3.00	28.60	1595.72
		4	0.24	277.04	39.80	334.32	15.86	37.40	399.30	1569.49	15.70	117.80	1290.98
		5	0.08	158.41	44.40	466.20	16.82	78.80	528.30	1510.68	29.40	108.30	999.55
30	4	1	0.00	9.90	4.80	78.72	1.21	1.00	0.00	0.45	1.00	0.00	0.52
		2	6.76	56.55	28.78	589.94	3.63	12.67	93.33	193.14	5.89	56.33	2207.11
		3	1.70	247.07	22.70	372.28	2.72	32.80	303.50	151.36	11.40	50.20	230.78
		4	0.12	106.65	28.30	580.15	2.96	31.00	227.50	84.25	7.40	19.90	44.15
		5	0.05	29.62	28.60	351.78	3.05	27.50	178.30	56.75	8.40	16.20	34.59
40	4	1	0.00	52.56	5.40	110.70	1.75	1.00	0.00	0.41	1.00	0.00	0.47
		2	2.68	252.50	26.56	435.51	8.43	5.33	71.67	813.41	1.67	11.11	1342.50
		3	6.13	313.12	46.80	575.64	14.35	48.20	770.80	1431.74	10.20	63.00	1375.56
		4	1.16	311.61	48.90	1002.45	15.91	56.60	625.50	808.86	13.20	56.90	505.49
		5	0.27	273.71	50.30	618.69	15.62	50.30	512.00	431.02	13.70	44.70	282.72
40	6	1	0.00	36.75	5.67	138.27	1.32	1.00	0.00	1.10	1.00	0.00	0.75
		2	5.53	186.32	23.50	573.40	5.29	4.50	32.20	83.42	3.60	27.50	2047.42
		3	3.51	314.17	31.70	773.48	4.58	75.60	390.40	199.67	10.10	55.20	294.09
		4	0.92	276.31	28.90	705.16	4.72	27.70	249.80	112.39	9.20	30.60	81.21
		5	0.06	83.56	30.20	921.10	4.57	29.10	223.50	89.62	7.80	21.20	49.18
40	8	1	1.38	41.62	4.70	190.35	0.50	1.00	0.00	0.80	1.00	0.00	0.98
		2	12.59	169.55	23.80	771.12	2.30	55.40	71.00	89.62	6.70	44.10	1139.39
		3	1.68	312.18	24.30	787.32	2.42	19.10	139.30	53.43	9.00	30.80	61.02
		4	0.58	136.89	19.80	801.90	1.93	20.10	135.80	54.75	7.80	22.40	41.00
		5	0.22	27.67	21.50	696.60	1.88	18.90	111.30	45.19	5.70	9.00	25.05

50	6	1	0.00	87.49	5.83	142.33	2.85	1.00	0.00	1.12	1.00	0.00	1.67
		2	0.00	391.37	37.60	688.08	14.39	5.60	90.60	982.64	1.60	16.40	1512.16
		3	5.72	344.28	54.60	1332.24	22.37	38.50	638.10	678.24	13.00	80.90	1089.50
		4	1.55	339.42	49.70	1212.68	20.88	43.40	532.20	491.67	12.40	52.00	383.52
		5	0.56	304.32	48.50	1183.40	19.53	35.60	402.20	278.38	10.60	38.20	166.17
50	8	1	0.00	96.37	5.00	202.50	2.18	1.00	0.00	1.67	1.00	0.00	2.18
		2	6.95	358.07	29.30	949.32	6.90	11.30	92.10	206.73	3.40	42.10	1802.78
		3	5.38	346.36	34.10	1381.05	7.86	28.30	383.00	225.71	10.40	58.70	287.03
		4	1.21	339.65	34.30	1389.15	7.89	22.30	244.90	167.09	9.10	30.40	125.68
		5	0.29	254.63	31.80	1287.90	7.48	24.40	229.20	146.16	7.60	22.20	87.61
50	10	1	0.00	137.99	4.10	165.64	0.90	1.00	0.00	1.60	1.00	0.00	1.66
		2	14.17	570.28	21.50	1085.75	4.01	9.70	43.80	91.49	6.70	69.20	1606.37
		3	4.22	335.47	25.00	1262.50	3.76	20.20	201.00	110.39	9.00	33.10	108.52
		4	0.43	280.36	24.90	1005.96	4.09	18.80	152.80	93.09	8.00	24.60	84.65
		5	0.23	125.38	23.70	957.48	3.62	20.20	135.10	93.24	6.20	14.00	54.67
Average		1	0.15	76.19	4.83	121.05	1.32	1.02	0.11	0.90	1.06	0.24	61.94
		2	5.81	267.40	24.84	594.19	6.59	12.19	56.61	350.05	4.19	34.47	1504.39
		3	3.43	292.85	33.72	765.51	8.42	33.52	346.88	522.94	9.89	48.23	595.96
		4	0.69	231.60	33.26	798.60	8.46	31.77	299.99	390.01	10.20	41.13	292.88
		5	0.20	140.78	33.64	748.12	8.25	34.68	270.07	302.64	10.98	32.54	193.85

6. Conclusion

The observed improvement in the positional model relative to the Manne model, and especially to the Wagner model, is because of the eliminated use of the big-M constant. Consequently, the model is linear and easier to solve (Chen & Powell, 1999a). In addition, reduction of the number of positions (Proposition 1) also helps reduce the need for computational effort.

Tight linear relaxation of the positional model enabled implementing the linear relaxation-based heuristics to generate the quality initial columns and more quickly solve the pricing problem. In solving the pricing problem, the heuristic methods (CH and LRBH_{PP}) were effective in reducing the need to use the exact method (PM_{PP}). This was proven when the exact method used fewer iterations. We also tested inserting only the optimal column or 12 best columns per iteration. However, the results were worse than those found by the proposed approach used.

The congestion level of production system affects the problem solving. This was proved by observing the results of the positional model for medium-size instances: as q increased, the performance improved. Therefore, we developed two methods for generating the initial solution for the HCG method: the ILS method for levels with less congestion (q = 1 and 2), and the LRBH_{IS} method for levels with greater congestion (q = 3, 4, and 5).

The HCG method found the optimal solution of 89 among 90 instances of congestion level q = 1. This is because the due dates were looser, which facilitated the ILS method in finding the optimal solution of 89 instances. The greatest performance difference between the HCG method and positional model occurred with a lower congestion level. For $q = 2$, the HCG method obtained the worst performance, and from $q = 3$, the performance improved when the congestion level increased until $q = 5$. Although the HCG method found fewer optimal solutions than the positional model, it had a smaller overall average gap than the positional model at 4.78% versus 34.78%, respectively.

Future work will involve determining a means to eliminate the negative effect that loose due dates have on decreasing the efficiency of the positional model. Other future work is to develop a re-optimization algorithm to reduce the computational cost of solving the pricing problem. This work would be similar to the Desrochers and Soumis (1988) proposed method, which reuses part of the solution of the preceding problem to reduce the cost of solving multiple times the shortest-path problem with the time-window problem. In addition, these approaches could be tested in other scheduling problems that use the big-M constant.

Acknowledgements

The authors would like to thank the National Council for Scientific and Technological Development (CNPq), Coordination of Personnel Improvement of Higher Education (CAPES), and Foundation for Research Support of the State of Minas Gerais (FAPEMIG) for their financial support.

References

Afzalirad, M., & Shafipour, M. (2015). Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions. *Journal of Intelligent Manufacturing*, doi:10.1007/s10845-015-1117-6.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, *187*(3), 985–1032.

Allahverdi, A., & Soroush, H. M. (2008). The significance of reducing setup times/setup costs. *European Journal of Operational Research*, *187*(3), 978–984 (2008)

Anghinolfi, D., & Paolucci, M. (2007). Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Computers & Operations Research*, *34*(11), 3471–3490.

Armentano, V. A., & de França Filho, M. F. (2007). Minimizing total tardiness in parallel machines scheduling with setup times: an adaptive memory-based GRASP approach. *European Journal of Operational Research*, *183*(1), 100–114.

Avalos-Rosales, O., Angel-Bello, F., & Alvarez, A. M. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *The International Journal of Advanced Manufacturing Tecnology*, *76*(9), 1705-1718.

Azizoglu, M., Çetinkaya, F. C., & Pamir, S. K. (2015). LP relaxation-based solution algorithms for the multi-mode project scheduling with a non-renewable resource. *European Journal of Industrial Engineering*, *9*(4), 450–469.

Bilge, U., Kirac, F., Kurtulan, M., & Pekgun, P. (2004). A tabu search algorithm for parallel machine total tardiness problem. *Computers & Operations Research*, *31*(3), 397–414.

Chang, P.-C., & Chen, S.-H. (2011). Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup time. *Applied Soft Computing*, *11*(1), 1263-1274.

Chen, C.-L. (2012). Iterated hybrid metaheuristic algorithms for unrelated parallel machines problem with unequal ready times and sequence-dependent setup times. *The International Journal of Advanced Manufacturing Tecnology*, *60*(5), 693–705.

Chen, C. L., & Chen, C. L. (2009). Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Tecnology*, *43*(1-2), 161–169.

Chen, J. F. (2009). Scheduling on unrelated parallel machines with sequence- and machine-dependent setup times and due-date constraints. *The International Journal of Advanced Manufacturing Tecnology*, *44*(11), 1204–1212.

Chen, Z.-L., & Lee, C.-Y. (2002). Parallel machine scheduling with a common due window. *European Journal of Operational Research*, *136*(3), 512–527.

Chen, Z. -L., & Powell, W. B. (1999a). Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing*, *11*(1), 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*(1), 220–232.

Chen, Z. -L., & Powell, W. B. (2003). Exact algorithms for scheduling multiple families of jobs on parallel machines. *Naval Research Logistics*, *50*(7), 823–840.

Cheng, T. C. E., Sin, C. C. S. (1990). A State-of-the-Art Review of Parallel-Machine Scheduling Research. *European Journal of Operational Research*, *47*(3), 271–292.

Dantzig, G. B., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, *8*(1), 101–111.

Desrochers, M., & Soumis, F. (1988). A reoptimization algorithm for the shortest path problem with time windows. *European Journal of Operational Research*, *35*(2), 242–254.

Du, J., & Leung, J. Y. T. (1990). Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, *15*(3), 483–495.

Figielska, E. (2009). A genetic algorithm and a simulated annealing algorithm combined with column generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources. *Computers & Industrial Engineering*, *56*(1), 142–151.

French, A. P., & Wilson, J. M. (2007). An LP-based heuristic procedure for the generalized assignment problem with special ordered sets. *Computers & Operations Research*, *34*(8), 2359–2369.

Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, *9*(6), 849–859.

Guedes, P. C., & Borenstein, D. (2015). Column generation based heuristic framework for the multiple-depotvehicle type scheduling problem. *Computers & Industrial Engineering*, *90*, 361–370.

Hardin, J. R., Nemhauser, G. L., & Savelsbergh, M. W. P. (2007). Analysis of bounds for a capacitated single-item lot-sizing problem. *Computers & Operations Research*, *34*(6), 1721–1743.

Hauge, K., Larsen, J., Lusby, R. M., & Krapper, E. (2014). A hybrid column generation approach for an industrial waste collection routing problem. *Computers & Industrial Engineering*, *71*, 10–20.

Houck, J. D. J., Picard, J. C., Queyranne, M., & Vemuganti, R. R. (1980). The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *Operations Research*, *17*, 93–109.

Jampani, J., & Mason, S. J. (2010). A column generation heuristic for complex job shop multiple orders per job scheduling. *Computers & Industrial Engineering*, *58*, 108–118.

Kim, D.-W., Kim, K.-H., Jang, W., & Chen, F. F. (2002). Unrelated Parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, *18*(3-4), 223–231.

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2013). Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources. *Flexible Services and Manufacturing Journal*, *25*(1), 25–47.

Kramer, H. H., Petrucci, V., Subramanian, A., & Uchoa, E. (2012). A column generation approach for power-aware optimization of virtualized heterogeneous server clusters. *Computers & Industrial Engineering*, *63*, 652–662.

Lange, J., & Werner, F. (2015). A comparison of approaches to modeling train scheduling problems as job-shops with blocking constraints. Tech. Rep. Preprints 2015-18, Otto-von-Guericke-University Magdeburg, Institute of Mathematical Optimization.

Lawler, E. L. (1977). A “pseudopolynomial” algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, *1*, 331–342.

Lee, Y. H., & Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, *100*(3), 464–474.

Lee, J.-H., Yu, J.-M., & Lee, D.-H. (2013). A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness. *The International Journal of Advanced Manufacturing Tecnology*, *69*(9), 2081–2089.

Li, K., & 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.

Lin, S.-W., Chou, S.-Y., & Ying, K.-C. (2007). A sequential exchange approach for minimizing earliness-tardiness penalties of single-machine scheduling with a common due date. *European Journal of Operational Research*, *177*(2), 1294-1301.

Lin, S.-W, Lu, C. C., & Ying, K.-C. (2011). Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints. *The International Journal of Advanced Manufacturing Tecnology*, *53*(1), 353–361.

Lin, S.-W, & Ying, K.-C. (2007). Solving single machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics. *International Journal of Advanced Manufacturing Technology*, *34*(11), 1183-1190.

Lopes, M. J. P., & de Carvalho, J. M. V. (2007). A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. *European Journal of Operational Research*, *176*(3), 1508-1527.

Lourenço, H. R., Martin, O., & Stützle, T. (2002). Iterated local search. In Handbook of Metaheuristics, F. Glover and G. Kochenberger, Eds. International Series in Operations Research & Management Science, vol. 57. Kluwer Academic Publishers, MA, USA, 321-353.

Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research*, *8*(2). 219-223.

M'Hallah, R., & Bulfin, R. L. (2005). Minimizing the weighted number of tardy jobs on parallel processors. *European Journal of Operational Research*, *160*(2), 471–484.

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

Nogueira, J. P. de C. M., Arroyo, J. E. C., Villadiego, H. M. M., & Gonçalves, L. B. (2014). Hybrid GRASP Heuristics to Solve an Unrelated Parallel Machine Scheduling Problem with Earliness and Tardiness Penalties. *Electronic Notes in Theoretical Computer Science*, *302,* 53–72.

Park, Y., Kim, S., & Lee, Y. H. (2000). Scheduling jobs on parallel machines applying neural network and heuristics rules. *Computers & Industrial Engineering*, *38*, 189–202.

Paula, M. R., Mateus, G. R., & Ravetti, M. G. (2010). A non-delayed relax-and-cut algorithm for scheduling problems with parallel machines, due dates and sequence-dependent setup times. *Computers & Operations Research*, *37*(5), 938–949.

Pessoa, A., Uchoa, E., Aragão, M. P. de, & Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, *2*(3), 259–290.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer.

Potts, C. N., & Kovalyov, M. Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, *120*(2), 228–249.

Rocha, P. L., Ravetti, M. G., Mateus, G. R., & Pardalos, P. M. (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers & Operations Research*, *35*(4), 1250–1264.

Sousa, J. P., & Wolsey, L. A. (1992). A time indexed formulation of non-preemptive single machine scheduling problems. Mathematical Programming, *54*(1), 353–367.

Thanh, P. N., Péton, O., & Bostel, N. (2010). A linear relaxation-based heuristic approach for logistics network design. *Computers & Industrial Engineering*, *59,* 964–975.

Vallada, E., Ruiz, R., & Minella, G. (2008). Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research*, *35*(4), 1350–1373.

van den Akker, J. M., Hoogeveen, J. A., & van de Velde, S. L. (1999). Parallel machine scheduling by column generation. *Operations Research, 47*(6), 862–872.

van den Akker, J. M., Hoogeveen, J. A., & van Kempen, J. W. (2012). Using column generation to solve parallel machine scheduling with minmax objective functions. *Journal of Scheduling*, *15*(6), 801–810.

Wagner, H. W. (1959). An integer linear-programming model for machine scheduling. *Naval Research Logistic Quarterly, 6*(2), 131–140.

Zeidi, J. R., & Hosseini, S. M. (2015). Scheduling unrelated parallel machines with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Tecnology*, *81*(9), 1487–1496.

Zhu, Z., & Heady, R. B. (2000). Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach. *Computers & Industrial Engineering*, *38*, 297–305.