Hindawi
Journal of Applied Mathematics
Volume 2017, Article ID 9452762, 10 pages
https://doi.org/10.1155/2017/9452762

Research Article

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

Francisco Regis Abreu Gomes[1] and Geraldo Robson Mateus[2]

[1]*Graduate Program in Production Engineering, Federal University of Minas Gerais and Federal Institute of Education, Science and Technology of Ceará, Belo Horizonte, MG, Brazil*
[2]*Computer Science Department, Federal University of Minas Gerais, Belo Horizonte, MG, Brazil*

Correspondence should be addressed to Francisco Regis Abreu Gomes; regisgomes@ifce.edu.br

Received 27 March 2017; Accepted 29 May 2017; Published 3 July 2017

Academic Editor: Dar-Li Yang

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.

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. de Paula et al. [7] proposed a nondelayed 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, de 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. Arnaout 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 multistart 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 ([8, 15–19]). 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 multicut 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:

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

$$\text{s.a:} \quad \sum_{i \in M} y_{ij} = 1, \quad \forall j \in N \tag{2}$$

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

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

$$\sum_{k \in N} x_{i0k} \leq 1, \quad \forall i \in M \tag{5}$$

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

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

$$C_{\max} \geq C_j, \quad \forall j \in N \tag{8}$$

$$C_0 = 0, \tag{9}$$

$$C_j \geq 0, \quad \forall j \in N \tag{10}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall j, k \in N_0,\ j \neq k,\ i \in M \tag{11}$$

$$y_{ij} \in \{0, 1\}, \quad \forall j \in N,\ i \in M. \tag{12}$$

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. However, 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 subcycles; that is, if $x_{ijk} = 1$ the completion time of job k should be greater than or equal to the completion time of job j, and if $x_{ijk} = 0$, these constraints become 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 nonnegativity 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 ([23–29]). 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. Côté 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 multicut technique [39]. Magnanti and Wong [40] defined the concept of Pareto-optimal cut for degenerate Benders subproblems and applied the multicut technique.

The proposed acceleration procedures are aimed at reducing the execution time of the master problem and multicut generation. Methods to improve the quality and quantity of cuts generated like Pareto-optimal cut [40], covering cut bundle [27], maximum feasible subsystem [23], and maximization density cut [28], among other methods cited, were not used because they depend on 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 a 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 is from Geoffrion and Graves [32], and second is 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 $\mathrm{LB}_1 \leq \mathrm{LB}_2 \leq \mathrm{LB}_3 \leq \cdots \leq \mathrm{LB}_k$. 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. Multicuts. 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 subcycles. For instance, consider six jobs labeled 1 to 6. Two subcycles would be 0-1-3-4-0 and 6-5-2-6. Tran and Beck [8] proposed the cut shown below.

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

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 $\max(s_{ikc}) \geq s_{ibc} + s_{icd} - s_{ibd}$, 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 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. By quality solutions, we mean those that have a value S, LB $\leq S \leq$ UB. The optimal solution of the next iteration may be among these solutions, if the method has not been terminated. Therefore, these solutions, including the optimal solution, are stored in a set called solutions pool. Each solution of the pool is solved by the subproblem, not just the optimal solution, which is why the procedure is a multicut. 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 multicut 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 ([42–44]). 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 subcycles, that is, constraints (7), and consequently constraints (8) and (10). For this reason, the master problem may find job sequences with subcycles, 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 subcycles.

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 subcycles 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

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

Algorithm 1: Subproblem evaluation.

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^h_i is generated and its completion time C^h_i is found. When $C^h_i > C^{mp}_i$, the sequence has a subcycle, so a CBC is generated and added to the master problem. The biggest C^h_i is the iteration makespan $C^h_{\max}$. If $C^h_{\max}$ 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.

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 multicut procedure outlined in Section 3.2. Algorithm 1 evaluates each one of the solutions. The ICBD algorithm is presented in Algorithm 2.

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 $\leq$ 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. The test instances obtained from Vallada and Ruiz (2011) are $N \in \{20, 30, 40, 50, 60\}$ 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. 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 Tran and Beck [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), which is calculated as $100 * (\text{opt} - LB_1)/LB_1$. 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. However 90% of instances are solved within two iterations, and the maximum number of iterations was 5 which occurred once. Therefore, with this instance set the ICBD method was performed using only the multicut procedure because other procedures only consume computational time and not bring any advantage. In the combinations 10×2, 10×3, 10×4, and 20×4 the ICBD method does not reduce the number of iterations or increase the number of cuts generated. In other combinations there

```
(1) begin
(2)  h ← 0; UB ← +∞; stop ← false;
(3)  while (stop = false) do
(4)   h ← 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
```

Algorithm 2: ICBD. *Restricted or original master problem.

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

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 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; the remainder of the table is divided into five groups, the first four groups show the results for four numbers 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% versus 0.16%, respectively. Gaps increase as the number of machines also increases, but the opposite occurs when increasing the number of jobs.

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

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		

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*

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. The ICBD method used 51.88% less runtime than the T&B method,

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	Original time
20	2	4.27	0.00	0.57	5.43	1.20	22.97	3.73	1.12	0.33
	3	5.17	0.00	0.53	4.87	1.50	33.73	7.60	1.97	0.75
	4	5.77	0.00	0.50	4.90	1.20	44.13	6.83	5.09	1.93
	5	5.13	0.00	0.43	5.07	1.73	20.80	5.37	2.22	6.44
30	2	8.07	0.00	1.07	5.80	1.83	31.47	8.10	3.59	1.18
	3	10.77	0.00	1.77	7.03	1.80	62.07	10.97	23.71	6.28
	4	8.80	0.00	1.13	6.20	1.33	66.63	8.07	46.64	59.3
	5	3.90	0.00	0.23	4.37	1.53	16.47	4.90	7.91	35.07
40	2	8.57	0.00	1.70	6.33	1.90	36.17	8.13	7.43	1.84
	3	11.70	0.00	1.53	7.60	1.23	67.93	5.77	44.02	7.75
	4	9.43	0.00	2.00	6.97	1.47	88.10	10.80	347.75	45.53
	5	5.90	0.00	0.97	5.37	2.20	29.00	8.03	55.59	369.05
50	2	13.63	0.00	3.70	7.20	3.90	45.10	21.83	17.25	3.4
	3	11.97	0.00	2.60	8.47	1.97	80.27	11.33	293.77	83.28
	4	12.27	0.00	3.40	8.43	1.67	116.30	12.97	841.21	303.00
	5	8.03	0.00	1.83	6.27	2.33	49.87	11.73	307.05	947.89
60	2	12.07	0.00	3.37	7.93	2.87	50.47	16.70	42.00	8.39
	3	18.17	0.00	5.17	10.73	2.73	107.07	17.40	328.53	188.97
	4	10.40	0.00	4.40	9.00	1.53	130.63	13.80	1185.92	443.66
	5	8.20	0.00	4.40	8.60	1.20	161.00	22.00	1802.03	1179.53
Average		9.11	0.00	2.07	6.83	1.86	63.01	10.80	268.24	184.68

higher value than that 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 obtain the optimal solution more difficultly. Therefore, it justifies the use of the three improvement procedures.

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 multicut 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.

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 multicut 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 of 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].

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to acknowledge Dr. Tony T. Tran for providing the instances used in Tran and 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, and A. Alvarez, "Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times," *International Journal of Advanced Manufacturing Technology*, vol. 76, no. 9-12, pp. 1705–1718, 2014.

[2] M. Afzalirad and 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.

[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 and 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, and J. Weglarz, *Scheduling Computer and Manufacturing Processes*, Springer Berlin Heidelberg, Berlin, Germany, 1996.

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

[7] M. R. de Paula, G. R. Mateus, and 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 and J. C. Beck, "Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups," in *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI '12)*, pp. 774–779, 2012.

[9] M. R. de Paula, M. n. Ravetti, G. R. Mateus, and 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, and K.-C. Ying, "Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints," *International Journal of Advanced Manufacturing Technology*, vol. 53, no. 1–4, pp. 353–361, 2011.

[11] E. Vallada and 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.-C. Ying, Z.-J. Lee, and S.-W. 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, and D.-H. Lee, "A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: Minimizing total tardiness," *International Journal of Advanced Manufacturing Technology*, vol. 69, no. 9-12, pp. 2081–2089, 2013.

[14] J.-P. Arnaout, R. Musa, and 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. An International Journal*, vol. 10, no. 4, pp. 385–401, 2005.

[16] J. N. Hooker, "An integrated method for planning and scheduling to minimize tardiness," *Constraints. An International Journal*, vol. 11, no. 2-3, 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 and 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 and J. N. Hooker, "Single-facility scheduling by logic-based Benders decomposition," *Annals of Operations Research*, vol. 210, pp. 245–272, 2013.

[20] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, pp. 238–252, 1962.

[21] J. N. Hooker and G. Ottosson, "Logic-based Benders decomposition," *Mathematical Programming, Series B*, vol. 96, no. 1, pp. 33–60, 2003.

[22] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

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

[24] H. D. Sherali and B. J. Lunday, "On generating maximal non-dominated Benders cuts," *Annals of Operations Research*, vol. 210, pp. 57–72, 2013.

[25] N. Papadakos, "Practical enhancements to the Magnanti-Wong method," *Operations Research Letters*, vol. 36, no. 4, pp. 444–449, 2008.

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

[27] G. K. Saharidis, M. Minoux, and 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. Saharidis and M. G. Ierapetritou, "Speed-up Benders decomposition using maximum density cut (MDC) generation," *Annals of Operations Research*, vol. 210, pp. 101–123, 2013.

[29] N. Azad, G. K. Saharidis, H. Davoudpour, H. Malekly, and 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, pp. 125–163, 2013.

[30] D. McDaniel and 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, and E. Jewkes, "Logic-based Benders decomposition for an inventory-location problem with service constraints," *Omega (United Kingdom)*, vol. 55, pp. 10–23, 2015.

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

[33] G. Côté and 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 and A. Lodi, "Local branching," *Mathematical Programming. A Publication of the Mathematical Programming Society*, vol. 98, no. 1-3, Ser. B, pp. 23–47, 2003.

[35] C. A. Poojari and 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 and 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, and 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, pp. 213–244, 2013.

[38] M. Jenabi, S. M. Fatemi Ghomi, S. A. Torabi, and S. H. Hosseinian, "Acceleration strategies of Benders decomposition for the security constraints power system expansion planning," *Annals of Operations Research*, vol. 235, pp. 337–369, 2015.

[39] F. You and I. E. Grossmann, "Multicut Benders decomposition algorithm for process supply chain planning under uncertainty," *Annals of Operations Research*, vol. 210, pp. 191–211, 2013.

[40] T. L. Magnanti and R. T. Wong, "Accelerating Benders decomposition: algorithmic enhancement and model selection criteria," *Operations Research. The Journal of the Operations Research Society of America*, vol. 29, no. 3, pp. 464–484, 1981.

[41] V. Jain and I. E. 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, and 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, and J. Wein, "Minimizing average completion time in the presence of release dates," *Mathematical Programming*, vol. 82, no. 1-2, Ser. B, pp. 199–223, 1998.

[44] J. M. van den Akker, C. P. van Hoesel, and M. W. Savelsbergh, "A polyhedral approach to single-machine scheduling problems," *Mathematical Programming. A Publication of the Mathematical Programming Society*, vol. 85, no. 3, Ser. A, pp. 541–572, 1999.

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