A Non-Delayed Relax-and-Cut Algorithm for Scheduling Problems with Parallel Machines, Due Dates and Sequence-Dependent Setup Times

Mateus Rocha de Paula[*,a,1], Geraldo Robson Mateus[a], Martín Gómez Ravetti[b]

[a] *Universidade Federal de Minas Gerais. Instituto de Ciências Exatas - Departamento de Ciência da Computação. Av. Antônio Carlos, 6627, Pampulha, Belo Horizonte/MG, Brazil - CEP 31270-901*

[b] *Centre for Bioinformatics, Biomarker Discovery, and Information-Based Medicine, School of Electrical Engineering and Computer Science, The University of Newcastle. University Drive, Callaghan, NSW 2308, Australia.*

Abstract

Consider the problem of scheduling a set of jobs to be processed exactly once, on any machine of a set of unrelated parallel machines, without preemption. Each job has a due date, weight, and, for each machine, an associated processing time and sequence-dependent setup time. The objective function considered is to minimize the total weighted tardiness of the jobs.

This work proposes a Non-Delayed Relax-and-Cut algorithm, based on a Lagrangean relaxation of a time indexed formulation of the problem. A Lagrangean heuristic is also developed to obtain approximate solutions.

Using the proposed methods, it is possible to obtain optimal solutions within reasonable time for some instances with up to 180 jobs and six machines. For the solutions for which it is not possible to prove optimality, interesting gaps are obtained.

Key words: Scheduling Problems, Parallel Machines, Lagrangean Relaxation, Relax-and-Cut

[*] Corresponding author

Email addresses: mateusrp@ufmg.br (Mateus Rocha de Paula), mateus@dcc.ufmg.br (Geraldo Robson Mateus), martin.ravetti@newcastle.edu.au (Martín Gómez Ravetti)

[1] Fax: +55 31 3409 4188
+61 2 4921-6929

1. Introduction

Consider a factory with a set of jobs that have to be processed, without preemption, on any machine of a set of unrelated parallel machines. Each job may have different processing times, and sequence-dependent setup times, for each machine.

The problem studied in this work is based on a real case of a particular plant of a refractory brick factory. It produced more than 8000 products, most of them with the same production sequence consisting in 5 stages: material input measurement, blending, conformation, heat treatment and packaging. In each of these stages, the plant has more than one machine, and most of the products can be processed by any machine at a particular stage.

In this scenario, the conformation stage is the most critical part of the production, and is closely related to the quality of the final product. Moreover, the production plan of the plant is made according to the conformation stage's planning. After the conformation stage's planning is defined, the other stages' planning are adjusted to meet its objectives. For this reason, most of the planning effort is invested on this stage. This work will focus on the conformation stage, which is modelled a Scheduling Problem with Unrelated Parallel Machines.

The conformation stage of the studied plant uses seven machines: four identical, which require manual work, two fully automatic and the last one designed for a specific kind of bricks. The production plan is made considering the unrelated parallel machines case with the first six machines and a single machine case for the seventh, both considering sequence-dependent setup times and due dates. A more detailed explanation can be found on [32].

Tanaka and Araki [37] propose a Branch-and-Bound algorithm, which uses a Lagrangean Relaxation to obtain lower bounds, to minimize the total tardiness on Scheduling Problems with Parallel Identical Machines. It can solve problems with up to 25 jobs and any number of identical machines. Liaw et al. [22] tackle the more general problem of minimizing the total weighted tardiness on Scheduling Problems with Unrelated Parallel Machines using lower bounds based on the assignment problem approximation. It can solve problems with up to 18 jobs and four machines.

Mokotoff and Chrétienne [26] developed a cutting plane algorithm for the minimization of the makespan on Scheduling Problems with Unrelated Parallel Machines. It can solve instances with up to 20 machines and 200 jobs. Mokotoff [25] uses a very similar approach, combined with pre-processing and other specific approaches to improve its previous results. It can solve instances with up to

100 machines and 1000 jobs.

Kedad-Sidhoum et al. [19] proposed lower bounds for the earliness-tardiness Scheduling Problem with Identical Machines with distinct due dates. Such lower bounds are assignment-based, generalized from lower bounds proposed for the single machine case. A time-indexed formulation is then investigated to derive efficient bounds through column generation or lagrangean relaxation. The gap between the proposed lower bounds and an upper bound based on a local search is about 1.5% for instances with up to 90 jobs and six machines.

J. Pereira Lopes and Valério de Carvalho [18] developed a Branch-and-Price algorithm to minimize the total weighted tardiness on Scheduling Problems with Unrelated Parallel Machines and Sequence-Dependent Setup Times, availability dates for the machines and due dates for the jobs. Their method used a "primal-box" technique and a specific branching variable selection rule to accelerate the column generation. It can solve instances with up to 150 jobs and 50 machines.

Rocha et al. [33] proposed two Mixed-Integer Programming (MIP) formulations to tackle Scheduling Problems with Unrelated Parallel Machines and Sequence Dependent Setup Times: one based on Manne [24] and another based on Wagner [38]. Ravetti [32] proposed also a Time-Indexed formulation that is explored in this work. Rocha et al. [33] propose a Branch-and-Bound algorithm based on constraint programming techniques, that used polynomial methods to quickly obtain lower-bounds. Ravetti [32]'s approaches can solve instances with up to 16 jobs and 6 machines, and Rocha et al. [33] can solve instances with up to 25 jobs and 6 machines, for the minimization of the makespan plus the total weighted tardiness.

See Li and Yang [21] for a survey on a few recent advances on models, relaxations and algorithms for minimization of the total weighted completion time on non-identical parallel machines.

For more information on time-indexed formulations for scheduling problems, refer to Sousa and Wolsey [36], Queyranne and Schulz [30], Akker et al. [2] and Akker et al. [1]. Refer also to Schulz [35] for a discussion about different mathematical formulation approaches to machine scheduling.

The scheduling problem with unrelated parallel machines, due dates and sequence-dependent setup times, tackled in this paper, is known to be difficult. It is also not addressed often in the literature considering unrelated parallel machines, sequence-dependent setup times and due dates all at once, which make the problem particularly complex. Moreover, by the time of this research, it was not possible to find benchmark instances available on the literature even for similar problems. Thus the instances used were generated using the same algorithm as in

Rocha et al. [33], considering a harder case than that found in the real case. The instances generated have six unrelated parallel machines, instead of the four identical and two unrelated, and admited planning horizons larger than those found in the real case. The instance generator and all necessary information can be found at the *UFMG Scheduling Group* home page.

It is also worth noticing that the difficulty of instances of this problem is not only measured by its size. In fact, when minimizing the total weighted tardiness, a large number of available machines (with respect to the number of jobs) usually makes the instance easier, since more jobs can be processed in parallel, which often leads to less tardy (and thus penalized) jobs. The realistic constraints, sequence-dependent setup times and due dates, also affect the instance complexity (see [33]).

In this work, three methods to obtain lower bounds for the problem are detailed. The first method is the linear programming relaxation of Integer Programming formulations of the problem. On Section 2, the linear programming relaxation of a time-indexed formulation, proposed by Ravetti [32], is compared with the linear programming relaxations of two other formulations, based on Wagner [38]'s and Manne [24]'s works. The second, is a lagrangean relaxation of the same time-indexed formulation. This method is previously suggested in Ravetti [32], but only preliminary results are presented. To address convergence issues of Ravetti [32]'s method, the third method is a novel Non-Delayed Relax-and-Cut algorithm, which is developed and presented on Section 3.3. On Section 3.4, a novel Lagrangean Heuristic, also based on Ravetti [32]'s lagrangean relaxation, is proposed. Finally, on Section 4 the developed algorithms are extensively tested, and the conclusions are summarized on Section 5.

1.1. Scheduling Problems with Parallel Machines and Sequence-Dependent Setup Times

Consider a set of unrelated machines $\mathscr{M} = \{1, 2, .., |\mathscr{M}|\}$ and a set of jobs $\mathscr{J} = \{1, 2, .., |\mathscr{J}|\}$.

The scheduling problems with parallel machines tackled in this work consist in assigning a machine for each job, and sequencing the set of jobs assigned to each machine considering the objective of minimizing the total weighted tardiness. Every job $j \in \mathscr{J}$ has exactly one associated operation, which can be performed by any machine $m \in \mathscr{M}$. Preemption is not allowed.

In the realistic problem studied in this work, positive weights w_j are associated to each job $j \in \mathscr{J}$. In order to process a job j immediately after a job j' on machine m, it is required a positive setup time $s_{j'jm}$, that depends both on the

sequence of jobs $j' \prec j$, job j being scheduled immediatly before another job j', and on the machine $m \in \mathscr{M}$ where they are processed; and a positive processing time p_{jm}, that depends only on machine $m \in \mathscr{M}$ where job j is to be processed.

Define the completion time of job j recursively as $C_j = C_{j'} + p_{jm} + s_{j'jm}$, where $C_{j'}$ is zero if job j is the first job scheduled or the completion time of the previous job $j' \prec j$ otherwise. The tardiness τ_j of each job is calculated as $\tau_j = max(C_j - d_j, 0)$, where d_j is the due date of job j. Let C_{max} be the completion time of the last job to finish processing ($C_{max} = max_{j \in \mathscr{J}}\{C_j\}$). All processing must be done within a stipulated time horizon $T > C_{max}$. In this work, $T = \infty$, unless explicitly told otherwise. It is assumed that all jobs and machines are available at the first time index. These problems are also known as $R|s_{jj'm}, \tilde{d}_j|\sum w_j\tau_j$.

For a review on scheduling problems, notation and classical approaches for them, refer to Pinedo [29], Lee and Pinedo [20], Blazewicz et al. [4] and Brucker [5].

Throughout this text, the following notation will be used:

- j: A particular job in $\mathscr{J}$.
- m: A particular machine in $\mathscr{M}$.
- T: The planning horizon. Consider that a planning horizon of size T have $T+1$ discrete units of time, that is, jobs can be scheduled at times $t = 0, ..., T$.
- t: A particular time period, $0 \leq t \leq T$.
- $|\mathscr{S}|$: The cardinality of set $\mathscr{S}$. That is, the number of jobs or machines, if $\mathscr{S} = \mathscr{J}$ or $\mathscr{S} = \mathscr{M}$, respectively.
- p_{jm}: Processing time of job j on machine m.
- $s_{jj'm}$: Setup time needed to process job j' immediately after job j on machine m.
- $\delta_{jj'm}$: $p_{jm} + s_{jj'm}$
- d_j: The due date of job j. The processing of job j has to finish until time $t = d_j$. Otherwise a penalty proportional to the job's weight and its tardiness, $w_j * \tau_j$, will be considered in the objective function.
- w_j: The weight of job j.
- C_j: The completion time of job j.
- C_m: The completion time of the last job processed by machine m.
- C_{max}: The greatest completion time of all jobs, $max_{j \in \mathscr{J}}\{C_j\}$.
- τ_j: The tardiness of job j. $\tau_j = max\{0, C_j - d_j\}$.
- M: a very large constant.

2. Mixed-Integer Programming Formulations

The formulations found on the literature to tackle machine scheduling problems vary mainly on the way they discretize the assignment of jobs. Usually, they use start/completion time variables, linear ordering variables, time-indexed variables and positional date variables (see [35]).

Several formulations for machine scheduling problems are found on the literature (see Sousa and Wolsey [36], Queyranne and Schulz [30], Akker et al. [2], Akker et al. [1] and Schulz [35]). However, only two formulations, proposed in [31], apply for the particular problem addressed in this paper, which considers unrelated parallel machines, due dates and sequence-dependent setup times: one based on Wagner [38]'s work and one based on Manne [24]'s work.

While the formulation based on Wagner [38]'s work uses discrete positions on machines, the formulation based on Manne [24]'s work is based on job precedence. The time-indexed formulation, detailed in Model 1, uses discrete time positions in a similar way to the formulation based on Wagner [38]'s work, but instead of using positions, which assume variable durations, it uses time units with a fixed minimum duration (Ravetti [32]). The studied plant works with a 30 minute time unit.

The time-indexed formulation requires a planning horizon to be specified, that is, an upper bound on the optimal solution's makespan. Thus, the proposed implementations of the proposed algorithms based on the time indexed formulation use the makespan of a known feasible solution as a planning horizon, even though there's no guarantee that the optimal solution's makespan will fit on it. Still, using this method to estimate the planning horizon is convenient, since the aim of the algorithms that require this parameter is to obtain lower bounds, and lower bounds obtained with a subestimated time horizon would still be valid. Moreover, since the size of the time-indexed formulation grows substantially as the planning horizon grows, it is crucial to keep it conveniently low. In the real case considered, the planning horizon is set to one month.

3. Lagrangean relaxation

The obtained Linear Programming relaxation bounds of the formulations based on Wagner's and Manne's works are typically too weak to be of practical interest. Thus, a better alternative would be to use the Linear Programming relaxation of

Model 1: Time-Indexed Formulation for the Scheduling Problem with Parallel Machines and Sequence-Dependent Setup-Times, considering the minimization of the Total Weighted Tardinesses. The time horizon considered (T) is the makespan of a known feasible solution.

$$minimize \sum_{j \in \mathscr{J}} w_j \tau_j \tag{1a}$$

subject to

$$\sum_{m \in \mathscr{M}} \sum_{t=0}^{T-p_{jm}} x_{jmt} = 1 \qquad \forall j \in \mathscr{J} \tag{1b}$$

$$x_{jmt} + \sum_{u=t}^{t+p_{jm}+s_{jj'm}-1} x_{j'mu} \leq 1 \qquad \forall j, j' \neq j \in \mathscr{J}, \forall m \in \mathscr{M}, \forall 0 \leq t \leq T - p_{jm} \tag{1c}$$

$$\tau_j \geq \sum_{m \in \mathscr{M}} \sum_{t=0}^{T-p_{jm}} (t + p_{jm}) x_{jmt} - d_j \qquad \forall j \in \mathscr{J} \tag{1d}$$

$$\tau_j \geq 0 \qquad \forall j \in \mathscr{J} \tag{1e}$$

$$x_{jmt} \in \{0,1\} \qquad \forall j \in \mathscr{J}, \forall m \in \mathscr{M}, \forall 0 \leq t \leq T - p_{jm} \tag{1f}$$

Decision Variables:
x_{jmt}: 1 if job j is scheduled at position t of machine m, and 0 otherwise.

Objective Function and Restrictions:
(1a): The objective function is to minimize the sum of weighted tardinesses.
(1b): Each job must be scheduled at one time index of exactly one machine.
(1c): If job j is scheduled at time period t, no other job j' can be scheduled on the next $p_{jm} + s_{jj'm}$ periods, which are reserved to the processing of job j and proper machine setup.
(1d): The tardiness of job j is greater than or equals to the difference between its start time plus its processing time and its due date.
(1e): The tardinesses must be positive.
(1f): The decision variable x is binary.

Model 2: Lagrangean Relaxation of the Time-Indexed Formulation, considering the minimization of the Total Weighted Tardinesses, and relaxing the precedence constraints.

$$minimize_{\lambda} \sum_{j\in\mathscr{J}} w_j\tau_j + \sum_{j\in\mathscr{J}} \sum_{\substack{j'\in\mathscr{J}\\ j'\neq j}} \sum_{m\in\mathscr{M}} \sum_{t=0}^{T-p_{jm}} \lambda_{jj'mt}(x_{jmt} + \sum_{u=t}^{t+p_{jm}+s_{jj'm}-1} x_{j'mu} - 1) \quad (2a)$$

subject to

$$\sum_{m\in\mathscr{M}} \sum_{t=0}^{T-p_{jm}} x_{jmt} = 1 \qquad \forall j \in \mathscr{J} \quad (2b)$$

$$\tau_j \geq \sum_{m\in\mathscr{M}} \sum_{t=0}^{T-p_{jm}} (t+p_{jm})x_{jmt} - d_j \qquad \forall j \in \mathscr{J} \quad (2c)$$

$$\tau_j \geq 0 \qquad \forall j \in \mathscr{J} \quad (2d)$$

$$x_{jmt} \in \{0,1\} \qquad \forall j \in \mathscr{J}, \forall m \in \mathscr{M}, \forall 0 \leq t \leq T - p_{jm} \quad (2e)$$

the time-indexed formulation. However, it requires a lot of memory space, since it uses a very large number of variables and constraints, which limits the size of the instances that can be tackled with this method. A classical approach to this problem is to consider the Lagrangean relaxation of the formulation, relaxing enough constraints to reduce the problem to a more reasonable size.

For more information about Lagrangean relaxations see Held and Karp [15], Held and Karp [16], Fisher [8], Wolsey [39], Guignard [11] and Guignard [12].

3.1. Relaxing the Precedence Constraints

One possible Lagrangean relaxation is achieved by relaxing the precedence Constraints (1c). Model 2 details such a Lagrangean relaxation. Another Lagrangean relaxation that relaxes the assignment constraints 1b would also be possible. However it would still require too much memory space, and is thus uninteresting for the pursued purposes.

Note that, once a job-schedule is defined, the tardinesses can be easily calculated. Since the triplet job, machine and time index are fixed, and the Lagrangean multipliers are inputs for the problem, it is possible to pre-calculate the costs of each job scheduling using Equation 3 from Model 3.

To calculate the cost of scheduling a job j at the time index t on a machine m,

assume that $x_{jmt} = 1$. The weighted tardinesses can be easily calculated, since it involves only fixed variables and constants: the time period t, the processing time p_{jm} of job j at machine m, and its due date d_j.

Since x_{jmt} is set to 1, the Lagrangean multipliers associated with jobs that the job scheduling affects are directly determined from Expression 2a.

Finally, each job $j' \neq j$ will add $\lambda_{j'jmu}$ to the objective function for each time index $u \leq t$ that would make its processing time overlap the processing time of j, even if $x_{jmu} = 0$. That is, for each $u = \{t - p_{j'm} - s_{j'jm} \ldots t\}$.

Using these costs, it is possible to rewrite Equations 2a-2e of Model 2 as Model 3. Note that constraints 2c and 2d model $max\{0, t + p_{jm} - d_j\}$. Thus, if only positive tardinesses are considered on the β costs, these constraints may be discarded.

The resulting subproblem is a multiple choice problem, that can be solved in $O(|\mathscr{M}||\mathscr{J}|T)$ time by inspection. Since an optimal solution is achived simply by choosing the smallest β (a job assignment) value for each job, it has the integrality property (see Ross and Soland [34], Guignard [11], Guignard [12]), so constraints 1f may also be discarded. For the same reason, it follows that the lower bounds obtained by such relaxation are exactly the same as those obtained by the linear programming relaxation (see Guignard [11], Guignard [12] and Wolsey [39]), and might require more computation time to be obtained, depending on the convergence of the methods used to solve the Lagrangean Dual. However , it requires much less memory than a linear programming relaxation because all data can be processed directly, and there's no need to store numerous variables and constraints, necessary to create the model, in memory.

3.2. The Subgradient Method

A method to solve the Lagrangean Dual without using a linear programming system is the Subgradient Method (SM). The proposed implementation follows the basic SM scheme proposed in Wolsey [39]. The Lagrangean multipliers (λ) and current lower-bound are initially set with zero values. The upper bound is defined by the objective function value of a known feasible solution. Then SM iterations are run until a stopping condition is met. The proposed implementation stops if all subgradient vector values are null, or either when the lower bound equals the upper bound or when the step becomes too small (smaller than 0.0001). On every SM iteration k, a Lagrangean relaxation is solved for the current λ values. The upper (UB) and lower (LB) bounds are updated, if necessary. Then, a step size is calculated with Equation 6 considering the new subgradient vector SG (see Equation 5). Finally, new Lagrangean multipliers λ are calculated with

Model 3: Pre-processed Lagrangean Relaxation of a Time-Indexed Formulation, considering the minimization of the Total Weighted Tardinesses, and relaxing the precedence constraints, using β costs.

$$\beta_{jmt} = \max\{0, t + p_{jm} - d_j\} w_j + \sum_{\substack{j' \in \mathscr{J} \\ j' \neq j}} (\lambda_{jj'mt} + \sum_{u=t-p_{j'm}-s_{j'jm}+1}^{t} \lambda_{j'jmu}) \quad (3)$$

$$minimize_{\beta,\lambda} \sum_{j \in \mathscr{J}} \sum_{m \in \mathscr{M}} \sum_{t=0}^{T-p_{jm}} (\beta_{jmt} x_{jmt} - \sum_{\substack{j' \in \mathscr{J} \\ j' \neq j}} \lambda_{jj'mt}) \quad (4a)$$

subject to

$$\sum_{m \in \mathscr{M}} \sum_{t=0}^{T-p_{jm}} x_{jmt} = 1 \qquad \forall j \in \mathscr{J} \quad (4b)$$

Equation 7, for the calculated step size. Every 50 iterations without improvement of the lower bound, the π value (see Equation 6), initially set to 2, is arbitrarily halved.

$$SG^k_{jj'mt} = x^k_{jmt} + \sum_{u=t}^{\min\{t+p_{jm}+s_{jj'm}-1, T-p_{jm}\}} x^k_{j'mu} - 1 \quad (5)$$

$$step^k = \frac{\pi * (\rho * UB - LB)}{||SG||^2} \quad (6)$$

$$\lambda^{k+1} = max\{0, \lambda^k + Step * SG^k\} \quad (7)$$

The subgradients SG, for the Lagrangean relaxation relaxing the Precedence Constraints, are calculated using Equation 5, and the relaxed problems are solved.

3.3. Non-Delayed Relax-and-Cut

With the Lagrangean relaxation of the precedence constraints of the time-indexed formulation, a huge number of constraints are relaxed, leading to a huge

number of nonzero entries in the sugradient vector SG of the SM. Thus, from equations 5 and 6, the subgradient norm $||SG||^2$ value becomes enormous, resulting in a very small step and bad convergence.

Relax-and-Cut algorithms, as defined in Lucena [23], attempt to improve Lagrangean bounds by dynamically strengthening relaxations with the introduction of valid constraints, where strengthening constraints may or may not be explicitly dualized.

As for any Lagrangean relaxation algorithm, regular Relax-and-Cut algorithms, like those developed by Gavish [9] and Escudero et al. [7], start with a relaxation of a given model where a set of complicating constraints is dualized, while the remaining constraints are kept. The algorithm then proceeds by solving the corresponding Lagrangean Dual Problem. Valid constraints that violate the Lagrangean Dual Problem solution are then identified and either dualized or kept. Either way, a new Lagrangean Dual Problem is formulated and solved. This procedure continues until a stopping criterion is reached.

Since the described method suggests that only *some* of the available strengthening constraints are dualized, it could be interesting to reduce the number of nonzero entries on the subgradient vector used by the Subgradient Method. However, since preliminary testing showed that the convergence of the proposed subgradient method can be quite slow, solving potentially many Lagrangean Dual Problems could be rather expensive.

Alternatively, Lucena [23] proposed modifications to the SM to handle a large number of relaxed constraints. The idea behind the proposed scheme, called *Non-Delayed Relax-and-Cut*, is to dualize constraints *on-the-fly*, that is, as they become violated. The main difference between the *Non-Delayed Relax-and-Cut* and regular Relax-and-Cut schemes is that the dualization of violated constraints is not delayed until the Lagrangean Dual is solved. That is, strengthening constraints are dualized after each Lagrangean *relaxation* Problem is solved, instead of after each Lagrangean *Dual* Problem is solved.

At any iteration k of the SM, the relaxed constraints can be classified into three sets: those violated by x^k, those that have nonzero Lagrangean multipliers λ^k associated with them, and the remaining constraints. Throughout this text these constraints are referred as *Currently Violated Active Set* (CA^k), *Previously Violated Active Set* (PA^k) and *Currently Inactive Set* (CI^k). Note that a constraint may be both on CA^k and PA^k simultaneously.

If no jobs are scheduled in the time interval considered by a constraint from equation 1c, its associated subgradient value is negative (-1). If only one job is scheduled on that interval, the associated subgradient value is null. In these

cases, the respective constraints are not violated. Violated constraints have more than one job scheduled on the time interval it considers, and thus have positive (≥ 1) subgradient values. It follows that constraints in CI^k do not contribute to the Lagrangean costs at the current iteration. However, they play a decisive role in determining the step size on iteration k, because their squared value is considered. Moreover, from equation 7, if the current Lagrangean multiplier associated with such a constraint is null, it will remain null at the end of the iteration, since the step is a positive real number and the associated subgradient value will be null or negative. Therefore, in order to deal with a great number of relaxed constraints in CI^k, the subgradient values $SG^k_{jj'mt}$ are arbitrarily set to zero whenever $SG^k_{jj'mt} \leq 0$ and $\lambda_{jj'mt} = 0$ (see Beasley [3] and Lucena [23]).

The described modification allows problems with a lot of constraints in CI^k to be handled. However, the studied problem also introduces a large number of constraints in $CA^k \setminus PA^k$, that is, currently violated constraints with null Lagrangean multipliers associated with them. These inequalities will become effectively dualized at the end of the $k-th$ SM iteration. In order to deal with an exceedingly large number of these constraints, Lucena [23] proposed that only one of those constraints should be effectively relaxed at each iteration of the SM. Since the dualization of one of the constraints in $CA^k \setminus PA^k$ does not affect directly more than one machine, the proposed implementation relaxes at most one maximal constraint per machine at each iteration of the Subgradient Method, and the other constraints have their subgradient entries arbitrarily set to zero, thus becoming, in effect, constraints in CI^k (see Lucena [23]).

In order to choose a maximal constraint to be dualized for a machine $m \in \mathscr{M}$, it is necessary to identify the violated constraints at the current solution. Clearly, violated constraints have positive subgradient entries. Preliminary tests showed a "very violated" and penalized constraint, that is, that starts much sooner than it should and have a large associated λ value, is usually a good choice to be penalized. Thus, following Lucena [23]'s suggestion, for every machine m, jobs j and j' and a time index t are chosen such that $(t - t' + p_{jm} + s_{jj'm}) * \lambda_{jj'mt}$ is maximal, considering that job j starts its processing at time index t and job j' starts its processing time at time index t'. In this case, $SG_{jj'mt}$ is left unchanged, and all other subgradient entries associated with machine m are set to zero.

It is possible to choose one such maximal constraint in $O(nT^2)$, with a greedy algorithm. Although this algorithm is polynomial, it may not be efficient if a large planning horizon is considered.

3.4. Lagrangean Heuristic

Note that, since jobs scheduled on the first time-index of a machine cannot be anticipated, solutions of relaxed problems with precedence constraints dualized tend to distribute the jobs on the first time index of the machines and the Lagrangean multipliers tend to separate pairs of jobs as much as possible (penalizing violated constraints, that is, pairs of jobs with overlapping processing time periods). Since the assignment constraints already guarantee that all jobs are scheduled exactly once, when a lower bound is strong, in this case, it should be close to feasibility, except for a few jobs that still have overlapping processing times, because of a machine or sequence misplacement.

Indeed, preliminary tests showed that, for many instances (mainly the easy ones), the obtained lower bounds proved the optimality of known feasible solutions or provided a very small gap.

Since the jobs scheduled with a large associated β value (see equation 3) have a greater impact on the objective function, a Lagrangean heuristic to achieves feasibility, without compromising the objective function value, would be to schedule, in a greedy constructive fashion, the jobs in decreasing order of associated β values (β_{jmt} such that $x_{jmt} = 1$ in the solution of the current relaxed problem).

Running a local search might also help, since it guarantees the resulting solution to be a local minimum with respect to the associated neighborhood. In the proposed implementation, a Variable Neighborhood Search (VNS) heuristic, proposed by de Paula [6], is used for this purpose. The VNS heuristic takes the solution obtained by the Lagrangean heuristic as its initial solution. It is well known that good initial solutions often helps to lead local-search based heuristics to good final solutions, so it is expected that the solutions obtained using this method will be better than those obtained with a VNS implementation that uses simpler methods to obtain initial solutions.

In order to minimize the computational overhead introduced by the Lagrangean heuristic, the proposed implementation of the subgradient method runs the algorithm every time the lower bound improves by 10% since the last run of the Lagrangean heuristic. Since preliminary tests showed that the β values and the relaxed problem solutions change relatively little from one iteration to the next, running the Lagrangean heuristic more frequently on the SM would be too costly and would lead to the same solution too often. It is expected that the use of a local search after each run of the Lagrangean heuristic, instead of running the Lagrangean heuristic more frequently, would lead to better final solutions.

In this text, the proposed implementation of the lagrangean heuristic, that is, the greedy constructive step that schedules the jobs in order of β values, followed

by a local search step, performed by de Paula [6]'s VNS implementation, will be referenced as Improved Lagrangean Heuristic (ILH).

3.5. The VNS Algorithm

This algorithm, proposed by de Paula [6], is a local search based algorithm designed to tackle large instances of Scheduling Problems with Paralellel Machines and Sequence Dependent Setup Times. It is based on Variable Neighborhood Search (VNS. See Hansen and Mladenovic [13], Hansen and Mladenovic [14], Glover and Kochenberger [10]).

The basic VNS scheme applies a shake procedure to the current solution, that depends on the currently selected neighborhood. de Paula [6]'s implementation performs k single job movements (moves a random job from its current position to another, on the same or on another machine), where k is the current neighborhood. If the best solution found so far is improved on the local search step, neighborhood 1 is selected. The current neighborhood is incremented otherwise.

de Paula [6]'s implementation uses a constructive heuristic based on the Nawaz-Enscore-Ham (NEH) algorithm [28] to create an initial solution. It sequentially places each job, in increasing order of due dates, at the best schedule avaiable on the partial schedule, considering all machines and avaiable positions. However, any other constructive method, such as heuristics based on those proposed by Nagano and Moccellin [27] or Hoon Lee and Pinedo [17], would suffice.

The local search used, also proposed by de Paula [6] is based on the union of a swap and insertion neighborhoods. It efficiently analyzes all pairwise job swaps (between jobs resident on the same or different machines) and all single job movements (change of positions also on the same machine or to another machine), accepting a better solution immediately (first-improvement local search). This procedure is repeated until no better solution is found.

4. Computational Tests

4.1. Instance Generation

Random instances are used in this work. They are generated with the same algorithm as in Rocha et al. [33]. Let h be the makespan (C_{max}) of a solution obtained by the *Earliest Due Dates* constructive heuristic, considering the job generation order as the job insertion order. The processing times, setup times, weights and due dates values are generated using a discrete uniform distribution on the ranges described in Table 1.

Table 1: Range values for the instance generation

Input data	Min	Max
Processing time	5	200
Setup time	25	50
Priority	1	3
Due date	$max_{j\in\mathscr{J}}(p_j)$	$\frac{2*h}{\theta}$

The setup times also satisfy the triangle inequality ($s_{ijm} \leq s_{ikm} + p_{km} + s_{kjm}, \forall i \neq j \neq k \in \mathscr{J}, m \in \mathscr{M}$).

As a rule of thumb, the greater the θ, the harder the instance is to solve, because the due dates are tighter and the scheduling system more congested.

For a fixed number of jobs, 40 instances are generated, 20 using $\theta = 1$ and 20 using $\theta = 5$. All instance information and the generator can be found at the *UFMG Scheduling Group* home page[2].

Since the aim of this work is to tackle large instances of the problem, and Rocha et al. [33] provides instances with only up to six machines and 25 jobs, new sets of instances were generated with more jobs. Moreover, even though [33] provide the algorithm of the instance generator, they do not provide the actual instances. For that reason, the instances generated with up to 25 jobs might be different from theirs.

4.2. Experiments and Results

The Lagrangean relaxations of the time indexed formulation, relax-and-cut algorithm and lagrangrean heuristic are implemented in C++ and compiled with the Intel C++ compiler (icpc) version 10.1.017, using the *-fast* flag only. The models of formulations based on Wagner [38]'s and Manne [24]'s works are implemented in C using the Ilog CPLEX Callable Library. The Time-Indexed formulation is implemented in C++ using the Ilog Concert Technology 2.4. All models are solved using Ilog CPLEX 10.2. All experiments are performed on a Intel Core 2 Quad 2.5GHz machine with 4GB RAM. All implementations and external processes (CPLEX, when applicable) are single-threaded.

In the following experiments, instances with 6, 8, 10, 12, 14, 15, 16, 20, 40, 80, 120, 140 and 180 jobs are considered. The number of machines is always fixed in six unrelated machines to match the real case, even though this is a slightly

[2]http://www2.dcc.ufmg.br/laboratorios/lapo/wiki/index.php/Scheduling_Instances

Table 2: Average Lower bounds (LB) obtained with the Linear Programming relaxation of formulations based on [24]'s (MLP) and [38]'s (WLP) works, proposed in [31] and with the time-indexed formulation (TLP), for problems minimizing the sum of weighted tardinesses. The exact results were also obtained with the time-indexed formulation.

Size (jobs)	**Exact**		**MLP**		**WLP**		**TLP**	
	Obj	**Time (s)**	**LB**	**Time (s)**	**LB**	**Time (s)**	**LB**	**Time (s)**
Instances with loose due dates								
6	39.9	1.87	0	0.01	0	0.01	34.45	0.29
8	51.25	16.86	0	0	0	0.02	39.3	2.78
10	58.4	56.68	0	0.01	0	0.03	43.35	9.83
12	70.55	143.63	0	0.01	0	0.06	60.2	23.95
14	42.75	386.57	0	0.01	0	0.09	36.9	62.34
16	61.95	991.95	0	0.01	0	0.17	50.95	447.52
Instances with tight due dates								
6	217.6	5.5	0	0.01	0	0.02	166.4	0.07
8	433.6	3.02	0	0.01	0	0.02	257.7	0.59
10	586.55	11.09	0	0.01	0	0.04	262.85	1.95
12	718.6	109.3	0	0.01	0	0.1	252.25	5.88
14	706.2	2522.82	0	0.01	0	0.11	264.25	16.54
16	677.25	3570.84	0	0.02	0	0.27	300	34.97

more general scenario, since the real case consists of four identical and two unrelated. A seventh machine is also present in the real case, but it is disconsidered for our purposes since it realizes only very specific jobs. 40 different instances are available for each size: 20 with loose due dates and 20 with tight due dates. Throughout this text, these sets of 40 instances are referenced by their size and a discrimination between the due date types is made whenever necessary. The same instances are used in different experiments, when the referenced size is the same.

4.2.1. Experiment 1

In this experiment, the linear relaxation of the considered models is solved for small instances, with 6, 8, 10, 12, 14 and 16 jobs. Table 2 depicts the results and compares them with exact results obtained solving the time-indexed formulation with the CPLEX 11.2 Optimization Package.

The average results presented on Table 2 show that, as expected, the linear programming relaxation of the time-indexed formulation is substantially superior to the others. In fact, the bounds obtained using the formulations based on Wagner's and Manne's works were the obvious null ones. It is worth noting that, although the obtained bounds are better, the CPU time needed to solve the linear relax-

Table 3: Average lower bounds for problems minimizing the total weighted tardiness obtained with the Basic Subgradient Method (BSM) and Non-Delayed Relax-and-Cut (NDRC).

Size (jobs)	**BSM**		**NDRC**	
	LB	**Time (s)**	**LB**	**Time (s)**
Instances with loose due dates				
10	42.05	11.77	44.30	9.69
20	64.95	273.87	67.05	253.00
40	58.80	999.38	58.70	998.42
80	56.60	2886.58	56.60	2886.56
Instances with tight due dates				
10	257.35	5.93	267.60	5.23
20	292.40	206.69	309.95	171.76
40	341.15	2983.22	333.10	2212.30
80	322.25	7200.45	320.20	7200.45

ation of the Time-Indexed formulation is often bigger than the CPU time needed to solve the linear relaxation of the other models, which is expected since the Time-Indexed formulation yields a bigger number of variables, constraints and data structures. Detailed results can be found on the *UFMG Scheduling Group* home page.

4.2.2. Experiment 2

In this experiment, the Non-Delayed Relax-and-Cut algorithm is compared to the Basic Subgradient Method, using sets of larger instances, with 10, 20, 40 and 80 jobs. The average results are presented on Table 3.

The average results presented on Table 3 suggest that, although the NDRC algorithm fixes the problem of the very small step size, convergence is still slow. In fact, preliminary tests showed that, at first, a lot of iterations of the NDRC algorithm produce very uninteresting bounds, but after some iterations (which may take a lot of time) an interesting direction is found and then convergence is much better. On a few instances, this makes a positive and significant difference, but the computation times are very close for most of the instances.

4.2.3. Experiment 3

This experiment aims to compare the proposed Improved Lagrangean Heuristic (ILH, see Section 3.4) with [6]'s VNS algorithm in terms of quality and computation time, to highlight their features for their due applications. Since the the

ILH algorithm provides a quality measure, which is not achieved by the VNS algorithm, it is expected that its computation time will be much greater. It is also expected that the quality of the solutions found by the ILH are at least as good as those found by the VNS algorithm because it uses a similar approach that considers also lagrangean information. Thus, it is important to notice that these algorithms may be useful in different situations and are, therefore, complementary in the sense that both would be interesting in practice. In a scenario where a solution does not have to be obtained instantly and small improvements on the solution may result in significant practical advantages, the ILH would be the best approach. In a scenario, where a solution has to be obtained very quickly and a quality measure is not so crucial, the pure VNS might be better.

Both algorithms are ran using the instances with 20, 40, 80 ,120, 140 and 180 jobs. The time limit is arbitrarily fixed at two hours. The gaps are calculated according to Equation 8, where the upper bound (UB) is the solution found by the heuristic and the lower bound (LB) is obtained using the NDRC algorithm. Since the pure VNS algorithm do not obtain lower bounds, the gap is calculated using the lower bound value obtained with the NDRC algorithm separatedly. The average results are presented on Tables 4 and 5. On both pure VNS implementation, and the search step of the lagrangean heuristic, iterations are run until the last five iterations fail to improve the current solution.

$$gap = \frac{UB - LB}{UB} \tag{8}$$

Table 4: Upper bounds for problems minimizing the total weighted tardiness on instances with loose due dates obtained with the Improved Lagrangean Heuristic (ILH). The upper bounds obtained using the ILH that are better than those obtained using VNS are in boldface. The gaps in italic indicate that the optimal solution is found.

		VNS			ILH		
Inst	**LowerBound**	**Sol**	**Gap**	**Time (s)**	**Sol**	**Gap**	**Time (s)**
			size = 20 jobs				
00	0	0	*0*	0.01	0	*0*	0.01
01	243	243	*0*	0.00	243	*0*	0.14
02	42	49	0.17	0.01	49	0.17	596.90
03	1	56	55.00	0.01	56	55.00	749.74
04	0	0	*0*	0.00	0	*0*	0.01
05	222	414	0.86	0.00	**410**	**0.85**	897.12

		VNS			ILH		
Inst	**LowerBound**	**Sol**	**Gap**	**Time (s)**	**Sol**	**Gap**	**Time (s)**
06	121	200	0.65	0.01	200	0.65	515.20
07	49	98	1.00	0.00	98	1.00	863.31
08	144	144	*0*	0.00	144	*0*	0.13
09	0	11	-	0.00	11	-	526.40
10	0	0	*0*	0.00	0	*0*	0.02
11	0	0	*0*	0.01	0	*0*	0.02
12	36	69	0.92	0.00	**36**	***0***	643.56
13	16	16	*0*	0.00	16	*0*	0.21
14	46	46	*0*	0.01	46	*0*	0.13
15	146	146	*0*	0.00	146	*0*	0.12
16	0	0	*0*	0.01	0	*0*	0.01
17	109	109	*0*	0.00	109	*0*	0.14
18	130	144	0.11	0.00	144	0.11	722.80
19	36	36	*0*	0.00	36	*0*	0.20
avg	67.05	89.05	3.09	0.00	87.20	**3.04**	275.81
			size = 40 jobs				
00	2	4	1.00	0.02	4	1.00	6599.37
01	128	128	*0*	0.01	128	*0*	1.80
02	21	51	1.43	0.02	51	1.43	5878.77
03	0	0	*0*	0.02	0	*0*	0.12
04	85	96	0.13	0.01	96	0.13	7200.00
05	0	0	*0*	0.01	0	*0*	0.12
06	32	32	*0*	0.00	32	*0*	1.94
07	105	105	*0*	0.00	105	*0*	2.29
08	170	250	0.47	0.00	250	0.47	7196.78
09	0	8	-	0.02	8	-	0.01
10	60	60	*0*	0.01	60	*0*	2.01
11	109	109	*0*	0.01	109	*0*	2.57
12	90	90	*0*	0.00	90	*0*	1.76
13	114	114	*0*	0.01	114	*0*	2.00
14	0	0	*0*	0.01	0	*0*	0.13
15	0	0	*0*	0.01	0	*0*	0.01
16	221	221	*0*	0.01	221	*0*	1.93
17	37	37	*0*	0.01	37	*0*	1.92
18	0	0	*0*	0.02	0	*0*	0.13
19	0	0	*0*	0.00	0	*0*	0.13

		VNS			ILH		
Inst	**LowerBound**	**Sol**	**Gap**	**Time (s)**	**Sol**	**Gap**	**Time (s)**
avg	58.70	65.25	0.16	0.01	65.25	0.16	1344.69
size = 80 jobs							
00	0	15	-	0.05	15	-	7200.02
01	0	0	*0*	0.04	0	*0*	0.92
02	24	24	*0*	0.03	24	*0*	20.07
03	14	55	2.93	0.04	**51**	**2.64**	7200.04
04	0	0	*0*	0.03	0	*0*	0.99
05	45	87	0.93	0.03	87	0.93	7200.02
06	0	0	*0*	0.03	0	*0*	0.92
07	0	0	*0*	0.03	0	*0*	0.91
08	94	94	*0*	0.04	94	*0*	19.54
09	87	87	*0*	0.04	87	*0*	19.24
10	57	106	0.86	0.06	106	0.86	7200.02
11	30	30	*0*	0.04	30	*0*	20.70
12	0	0	*0*	0.02	0	*0*	0.95
13	324	333	0.03	0.06	333	0.03	7200.02
14	15	15	*0*	0.05	15	*0*	20.63
15	33	33	*0*	0.08	33	*0*	19.24
16	102	172	0.69	0.04	172	0.69	7200.02
17	0	0	*0*	0.03	0	*0*	0.89
18	68	88	0.29	0.04	88	0.29	7200.03
19	239	301	0.26	0.03	301	0.26	7200.03
avg	56.60	72.00	0.32	0.04	71.80	**0.30**	2886.26
size = 120 jobs							
00	60	60	*0*	0.13	60	*0*	69.21
01	0	0	*0*	0.13	0	*0*	2.95
02	339	458	0.35	0.13	**441**	**0.30**	7200.13
03	0	0	*0*	0.11	0	*0*	3.14
04	73	183	1.51	0.18	183	1.51	7200.06
05	0	0	*0*	0.10	0	*0*	2.83
06	0	0	*0*	0.14	0	*0*	3.26
07	0	0	*0*	0.14	0	*0*	3.17
08	18	18	*0*	0.10	18	*0*	70.89
09	0	0	*0*	0.14	0	*0*	3.05
10	24	24	*0*	0.12	24	*0*	67.20
11	27	27	*0*	0.13	27	*0*	71.39

		VNS			ILH		
Inst	**LowerBound**	**Sol**	**Gap**	**Time (s)**	**Sol**	**Gap**	**Time (s)**
12	0	0	*0*	0.12	0	*0*	3.17
13	23	23	*0*	0.10	23	*0*	68.42
14	0	0	*0*	0.14	0	*0*	3.38
15	139	139	*0*	0.14	139	*0*	96.49
16	0	0	*0*	0.15	0	*0*	8.83
17	87	95	0.09	0.13	95	0.09	7200.12
18	202	370	0.83	0.18	370	0.83	7200.07
19	93	121	0.30	0.12	121	0.30	7200.13
avg	54.25	75.90	0.15	0.13	75.05	0.15	1823.89
size = 140 jobs							
00	9	9	*0*	0.21	9	*0*	132.65
01	56	81	0.45	0.18	**72**	**0.29**	7200.40
02	69	69	*0*	0.16	69	*0*	139.68
03	91	115	0.26	0.21	115	0.26	7200.23
04	30	30	*0*	0.20	30	*0*	156.11
05	61	78	0.28	0.30	78	0.28	7200.17
06	0	0	*0*	0.15	0	*0*	13.55
07	98	143	0.46	0.16	143	0.46	7200.21
08	32	182	4.69	0.20	182	4.69	719.77
09	47	47	*0*	0.21	47	*0*	110.00
10	0	0	*0*	0.19	0	*0*	11.22
11	4	40	9.00	0.15	40	9.00	7200.15
12	212	238	0.12	0.28	238	0.12	1926.10
13	33	33	*0*	0.19	33	*0*	128.79
14	57	180	2.16	0.21	180	2.16	675.77
15	1	60	59.00	0.29	60	59.00	178.19
16	72	84	0.17	0.16	84	0.17	599.33
17	53	53	*0*	0.20	53	*0*	140.73
18	0	0	*0*	0.22	0	*0*	0.63
19	0	0	*0*	0.22	0	*0*	0.53
avg	46.25	72.10	3.83	0.20	71.65	**3.82**	2046.71
size = 180 jobs							
00	0	0	*0*	0.46	0	*0*	1.21
01	0	0	*0*	0.44	0	*0*	1.10
02	0	0	*0*	0.38	0	*0*	1.10
03	0	0	*0*	0.53	0	*0*	1.14

		VNS			ILH		
Inst	**LowerBound**	**Sol**	**Gap**	**Time (s)**	**Sol**	**Gap**	**Time (s)**
04	12	12	*0*	0.34	12	*0*	180.51
05	0	0	*0*	0.41	0	*0*	1.18
06	0	0	*0*	0.43	0	*0*	1.23
07	213	213	*0*	0.46	213	*0*	179.72
08	24	24	*0*	0.38	24	*0*	179.72
09	36	36	*0*	0.44	36	*0*	178.18
10	0	0	*0*	0.38	0	*0*	0.93
11	0	0	*0*	0.37	0	*0*	0.84
12	28	28	*0*	0.48	28	*0*	179.49
13	0	0	*0*	0.42	0	*0*	1.02
14	0	0	*0*	0.44	0	*0*	1.09
15	114	140	0.23	0.60	140	0.23	875.52
16	82	282	2.44	0.63	282	2.44	826.89
17	0	0	*0*	0.56	0	*0*	1.15
18	35	35	*0*	0.33	35	*0*	184.70
19	0	0	*0*	0.38	0	*0*	1.05
avg	27.20	38.50	0.13	0.44	38.50	0.13	139.89

Table 5: Upper bounds for problems minimizing the total weighted tardiness on instances with tight due dates obtained with the Improved Lagrangean Heuristic (ILH). The upper bounds obtained using the ILH that are better than those obtained using VNS are in boldface. The gaps in italic indicate that the optimal solution is found.

		VNS			ILH		
Inst	**LowerBound**	**Sol**	**Gap**	**Time (s)**	**Sol**	**Gap**	**Time (s)**
size = 10 jobs							
00	155	457	1.95	0.02	**455**	**1.94**	7.72
01	137	364	1.66	0.00	364	1.66	4.93
02	512	854	0.67	0.00	854	0.67	5.15
03	352	611	0.74	0.01	611	0.74	5.35
04	331	669	1.02	0.00	669	1.02	3.21
05	191	682	2.57	0.00	682	2.57	5.46
06	383	849	1.22	0.01	849	1.22	8.97
07	301	712	1.37	0.00	712	1.37	4.64
08	191	572	1.99	0.01	572	1.99	4.25
09	195	424	1.17	0.00	424	1.17	1.21
10	360	691	0.92	0.00	691	0.92	2.93

		VNS			ILH		
Inst	**LowerBound**	**Sol**	**Gap**	**Time (s)**	**Sol**	**Gap**	**Time (s)**
11	188	481	1.56	0.00	481	1.56	1.75
12	411	830	1.02	0.00	830	1.02	3.34
13	222	491	1.21	0.00	491	1.21	5.79
14	133	632	3.75	0.00	**494**	**2.71**	4.17
15	284	622	1.19	0.00	622	1.19	4.77
16	227	454	1.00	0.00	454	1.00	5.39
17	333	675	1.03	0.00	675	1.03	3.68
18	292	582	0.99	0.00	582	0.99	15.18
19	154	538	2.49	0.00	538	2.49	4.86
avg	267.60	609.50	1.48	0.00	602.50	**1.42**	5.14
size = 15 jobs							
00	425	1305	2.07	0.01	1305	2.07	46.31
01	220	952	3.33	0.00	952	3.33	27.56
02	257	903	2.51	0.00	903	2.51	42.61
03	179	1286	6.18	0.01	925	**4.17**	79.70
04	388	1481	2.82	0.01	1481	2.82	36.75
05	131	1022	6.80	0.00	1022	6.80	47.75
06	375	1314	2.50	0.00	1314	2.50	26.39
07	297	1215	3.09	0.00	1215	3.09	48.41
08	257	1065	3.14	0.01	955	**2.72**	29.29
09	312	1014	2.25	0.00	1014	2.25	59.78
10	325	981	2.02	0.00	981	2.02	72.08
11	150	1359	8.06	0.00	1359	8.06	107.90
12	140	728	4.20	0.00	728	4.20	25.72
13	110	635	4.77	0.00	635	4.77	50.07
14	317	1174	2.70	0.00	1174	2.70	35.96
15	353	1191	2.37	0.01	1191	2.37	54.19
16	153	923	5.03	0.00	923	5.03	43.20
17	396	1267	2.20	0.01	1267	2.20	6.95
18	343	1242	2.62	0.01	1242	2.62	34.69
19	387	1351	2.49	0.00	1351	2.49	34.70
avg	275.75	1120.40	3.56	0.00	1096.85	**3.44**	45.50
size = 20 jobs							
00	149	1512	9.15	0.00	1512	9.15	199.41
01	447	1811	3.05	0.00	1811	3.05	158.98
02	271	1704	5.29	0.01	**1684**	**5.21**	218.88

			VNS			ILH	
Inst	**LowerBound**	**Sol**	**Gap**	**Time (s)**	**Sol**	**Gap**	**Time (s)**
03	383	1904	3.97	0.00	1904	3.97	166.51
04	80	1801	21.51	0.01	**1662**	**19.77**	162.35
05	536	2021	2.77	0.00	**2000**	**2.73**	237.09
06	525	2155	3.10	0.01	**2148**	**3.09**	270.08
07	356	1878	4.28	0.00	**1775**	**3.99**	290.97
08	324	2045	5.31	0.00	**2001**	**5.18**	77.35
09	282	1475	4.23	0.01	1475	4.23	112.67
10	214	1749	7.17	0.00	1749	7.17	139.02
11	327	1717	4.25	0.00	1717	4.25	208.91
12	283	1566	4.53	0.01	1566	4.53	107.34
13	166	1372	7.27	0.00	1372	7.27	112.24
14	108	1581	13.64	0.00	1581	13.64	91.21
15	246	1463	4.95	0.02	**1436**	**4.84**	147.71
16	340	2144	5.31	0.00	2144	5.31	125.51
17	248	1946	6.85	0.01	1946	6.85	130.02
18	392	2118	4.40	0.00	2118	4.40	230.34
19	522	2176	3.17	0.00	2176	3.17	142.69
avg	309.95	1806.90	6.21	0.00	1788.85	**6.09**	166.46
			size = 40 jobs				
00	348	5961	16.13	0.03	**5835**	**15.77**	2557.14
01	250	4880	18.52	0.02	**4837**	**18.35**	1958.05
02	196	5163	25.34	0.03	5163	25.34	2094.71
03	446	8396	17.83	0.03	8396	17.83	2263.85
04	539	6335	10.75	0.02	6335	10.75	2490.65
05	118	4449	36.70	0.02	4449	36.70	2142.60
06	107	5281	48.36	0.03	5281	48.36	1736.59
07	390	5952	14.26	0.02	**5915**	**14.17**	2629.80
08	422	5368	11.72	0.01	5368	11.72	2037.68
09	285	5569	18.54	0.04	5569	18.54	1792.25
10	399	5596	13.03	0.02	5596	13.03	2141.55
11	600	6797	10.33	0.02	**6771**	**10.29**	2813.48
12	194	5523	27.47	0.02	**5447**	**27.08**	1809.16
13	404	5331	12.20	0.01	**5008**	**11.40**	2240.56
14	383	6067	14.84	0.04	**5969**	**14.58**	2113.09
15	227	6146	26.07	0.02	6146	26.07	1951.00
16	540	7846	13.53	0.01	7846	13.53	2514.88

			VNS			ILH	
Inst	**LowerBound**	**Sol**	**Gap**	**Time (s)**	**Sol**	**Gap**	**Time (s)**
17	281	5786	19.59	0.02	5786	19.59	2336.94
18	142	4840	33.08	0.02	4840	33.08	1599.74
19	391	7814	18.98	0.01	7814	18.98	2155.16
avg	333.10	5955.00	20.36	0.02	5918.55	**20.26**	2168.94
			size = 80 jobs				
00	278	18238	64.60	0.37	**18217**	**64.53**	7200.08
01	161	16689	102.66	0.25	16689	102.66	7200.07
02	327	19694	59.23	0.23	**19656**	**59.11**	7200.08
03	322	19409	59.28	0.24	**19365**	**59.14**	7200.06
04	103	20445	197.50	0.28	**20404**	**197.10**	7200.08
05	260	17416	65.98	0.20	17416	65.98	7200.07
06	80	19356	240.95	0.15	**19332**	**240.65**	7200.12
07	131	21949	166.55	0.15	**21781**	**165.27**	7200.13
08	376	22479	58.78	0.31	22479	58.78	7200.06
09	260	19422	73.70	0.19	**19311**	**73.27**	7200.15
10	474	18530	38.09	0.18	18530	38.09	7200.07
11	601	19950	32.19	0.19	**19424**	**31.32**	7200.18
12	125	15517	123.14	0.13	15517	123.14	7200.08
13	661	17427	25.36	0.12	17427	25.36	7200.08
14	382	20152	51.75	0.24	**20097**	**51.61**	7200.09
15	412	20611	49.03	0.52	20611	49.03	7200.07
16	290	24477	83.40	0.15	24477	83.40	7200.06
17	172	22299	128.65	0.65	**22221**	**128.19**	7200.18
18	252	17721	69.32	0.34	17721	69.32	7200.09
19	737	22089	28.97	0.32	**21974**	**28.82**	7200.20
avg	320.20	19693.50	85.96	0.26	19632.45	**85.74**	7200.10

The results presented on Tables 4 and 5 show that the Lagrangean heuristic obtains good feasible solutions using the solutions obtained by the relaxed problems. Indeed, it can obtain optimal solutions for many instances, some with up to 180 jobs. It is noteworthy that the lagrangean heuristic is particularly interesting for the difficult instances, with tight due dates, improving many of the pure VNS solutions. Moreover, as expected, the obtained solutions are better on average than the solutions obtained with the pure VNS implementation. Also as expected, substantially more computation time is needed to obtain lower bounds than to obtain good feasible solutions.

The gaps obtained for the instances with tigh due dates are high, which indicate either that the proposed heuristics might still be far from the optimal solution,

or that the obtained lower bounds are poor for this class of instances. Either way, the Lagrangean heuristic is often able to obtain smaller gaps than the pure VNS heuristic.

5. Concluding Remarks

In this work, a Time-Indexed formulation is proposed to tackle scheduling problems with unrelated parallel machines, due dates and sequence-dependent setup times. Its linear relaxation provides much better bounds than the linear programming relaxation of the other formulations found on the literature, at the expense of a great memory usage.

A Lagrangean relaxation is proposed to obtain lower bounds for the problem. To deal with convergence problems encountered, a Non-Delayed Relax-and-Cut algorithm is also proposed.

Finally, a lagrangean heuristic is proposed to obtain approximate solutions.

The obtained bounds are particularly good for instances with loose due dates, proving optimality of known feasible solutions in several cases. For the other instances, the obtained results are still significant and better than the lower bounds obtained with the linear relaxation of other formulations found on the literature, even though they are still high.

As expected, because of the convergence of the methods used to solve the Lagrangean dual, the developed algorithm is often slower than linear programming approaches. However, the Lagrangean relaxation algorithm requires substantially less memory to run, and thus can obtain valid bounds even for large instances of the problem.

The Lagrangean heuristic successfully obtained good approximate feasible solutions. Indeed, it could obtain optimal solutions for many instances, some with up to 180 jobs. Moreover, the obtained solutions are better on average than the solutions obtained with the pure VNS implementation. The Lagrangean heuristic is particularly interesting for difficult instances, with tight due dates, obtaining results at least as good as those obtained by the pure VNS heuristic, and improving many of them. The improvement observed, however, is not statistically significant.

Even though substantially more computation time is needed to obtain lower bounds than to obtain good feasible solutions, this extra computational effort is advantageous for the real case considered because there is a quality improvement on the solutions, a quality measure and optimality proof (whenever possible). The company can use up to three days to finish a whole month's planning, which is still

much more than the lagrangean heuristic usually needs, even when improved by the VNS heuristic. Because of its speed advantage, [6]'s implementation, could still be a good alternative to quickly obtain solutions that are usually good, but without any measure of quality (e.g.: for a quick analysis of viability of a change of input data, such as a due date, or weight).

Since the main problem observed is related to memory issues, it is worth noticing that the ranges for random number generation are set to large values (up to 250), to reflect a more realistic situation and to experiment with a more difficult test case. That leads to a big planning horizon, and thus to huge data structures in terms of memory usage. The definition of these parameters depend on the application. If reduced values can be used for the setup and processing times, the proposed methods will be able to tackle much larger instances. In the real case considered, the planning horizon is set to one month, or 1440 time units, with a 30 minutes each. Since a typical planning horizon for an instance with six unrelated machines and 180 jobs have about 4500 time units, the proposed methods are sufficient satisfy the real case's requirements.

Further work will include: the development of a local search procedure for the Lagrangean heuristic that uses more information from the relaxed problem solution to guide the search; a deeper study of criteria to choose constraints to be penalized on the non-delayed relax-and-cut algorithm; research on other strong valid cuts to be used on a delayed relax-and-cut or branch-and-cut scheme; and specific methods that do not require a linear programming solver to solve the Lagrangean relaxation with assignment restrictions relaxed. Bound strengthening techniques, such as relaxation Linearization Technique (RLT), could also be used on the time-indexed formulation to improve its bounds, allowing for the proposed Relax-and-Cut algorithm to obtain stronger bounds. The study of Surrogate Constraints could also lead to interesting methods. Finally, parallel approaches for the problem might also be of interest to solve even bigger instances.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable feedback. This project was partially financed by the National Council for Scientific and Technological Development (CNPq).

References

[1] Akker, J. M. V. D., Hoesel, C. P. M. V., Savelsbergh, M. W. P., 1999. A polyhedral approach to single-machine scheduling problems. Mathematical

Programming 85, 541–572.

[2] Akker, J. M. V. D., 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.

[3] Beasley, J. E., 1993. Modern Heuristic Techniques for Combinatorial Problems. John Wiley & Sons, Inc., Ch. Lagrangean Relaxation, pp. 243–303, iSBN:0-470-22079-1.

[4] Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., Weglarz, J., 1996. Scheduling Computer and Manufacturing Processes. Springer - Verlag, Berlin.

[5] Brucker, P., 2004. Scheduling Algorithms. Springer-Verlag, Berlin.

[6] de Paula, M. R., January 2009. Heuristics for the minimization of the total weighted tardiness on scheduling problems with parallel machines and sequence-dependent setup times. Master's thesis, Federal University of Minas Gerais.

[7] Escudero, L. F., Guignard, M., Malik, K., December 1994. A lagrangian relax-and-cut approach for the sequential ordering problem with precedence relationships. Annals of Operations Research 50 (1), 219–237.

[8] Fisher, M. L., 1981. The lagrangian relaxation method for solving integer programming problems. Management Science 27, 1–18.

[9] Gavish, B., December 1985. Augmented lagrangean based algorithms for centralized network design. IEEE Transactions on Communications 33 (12), 1247– 1257.

[10] Glover, F., Kochenberger, G. A., 2002. Handbook of Metaheuristics. Kluwer Academics.

[11] Guignard, M., 1998. Efficient cuts in lagrangean 'relax-and-cut' schemes. European Journal of Operational Research 105 (1), 216–223.

[12] Guignard, M., 2003. Lagrangean relaxation. TOP: Sociedad de Estatistica e Investigación Operational 11 (2), 151–228.

[13] Hansen, P., Mladenovic, N., 1999. Variable neighborhood search: Principles and applications. European Journal of Operational Research 130 (3), 449–467.

[14] Hansen, P., Mladenovic, N., 2002. Variable neighborhood search. In: Pardalos, P. M., Resende, M. G. C. (Eds.), Handbook of Applied Optimization. Oxford University Press, New York.

[15] Held, M., Karp, R. M., 1970. The traveling-salesman problem and minimum spanning trees. Operations Research 18, 1138–1162.

[16] Held, M., Karp, R. M., 1971. The traveling-salesman problem and minimum spanning trees: Part ii. Mathematical Programming 1 (1), 1436–4646.

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

[18] J. Pereira Lopes, M., Valério de Carvalho, J. M., 2007. A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. European Journal of Operational Research 178 (3), 1508–1527.

[19] Kedad-Sidhoum, S., Solis, Y. R., Sourd, F., 2008. Lower bounds for the earliness–tardiness scheduling problem on parallel machines with distinct due dates. European Journal of Operational Research 189, 1305–1316.

[20] Lee, C.-Y., Pinedo, M., 2002. Optimization and heuristics of scheduling. In: Pardalos, P. M., Resende, M. G. C. (Eds.), Handbook of Applied Optimization. Oxford University Press, New York.

[21] 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, 2145–2158.

[22] Liaw, C.-F., Lin, Y.-K., Cheng, C.-Y., Chen, M., 2003. Scheduling unrelated parallel machines to minimize total weighted tardiness. Computers & Operations Research 30, 1777–1789.

[23] Lucena, A., 2005. Non delayed relax-and-cut algorithms. Annals of Operations Research 140 (1), 375–410.

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

[25] Mokotoff, E., 2004. An exact algorithm for the identical parallel machine scheduling problem. European Journal of Operational Research 152, 758–769.

[26] Mokotoff, E., Chrétienne, P., 2002. A cutting plane algorithm for the unrelated parallel machine scheduling problem. European Journal of Operational Research 141, 515–525.

[27] Nagano, M., Moccellin, J., 2002. A high quality solution constructive heuristic for flow shop sequencing. Journal of the Operational Research Society 53, 1374–1379.

[28] Nawaz, M., Enscore, E. E., Ham, I., 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11 (1), 91–95.

[29] Pinedo, M., 1995. Scheduling Theory, Algorithm and System. Prentice Hall.

[30] Queyranne, M., Schulz, A. S., 1994. Polyhedral approaches to machine scheduling. Preprint 408/1994, Dept. of Mathematics, Technical University of Berlin.

[31] Ravetti, M. G., 2003. Scheduling problems with parallel machines and sequence-dependent setup times. Master's thesis, Universidade Federal de Minas Gerais.

[32] Ravetti, M. G., 2007. Algorithms for a scheduling problem with parallel machines and sequence dependent setups. Ph.D. thesis, Universidade Federal de Minas Gerais.

[33] 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.

[34] Ross, G. T., Soland, R. M., December 1975. A branch and bound algorithm for the generalized assignment problem. Mathematical Programming 8 (1), 91–103.

[35] Schulz, A. S., 1996. Polytopes and scheduling. Ph.D. thesis, Technischen Universität Berlin.

[36] Sousa, J. P., Wolsey, L. A., 1992. A time indexed formulation of non-preemptive single-machine scheduling problems. Mathematical Programming 52, 353–367.

[37] Tanaka, S., Araki, M., 2008. A branch-and-bound algorithm with lagrangian relaxation to minimize total tardiness on identical parallel machines. International Journal of Production Economics 113, 446–458.

[38] Wagner, H. M., 1959. An integer linear-programming model for machine scheduling. Naval Research Logistics Quarterly 6 (2), 131.

[39] Wolsey, L. A., 1998. Integer Programming. Wiley.