Mixed integer programming formulations for a non-preemptive parallel machine scheduling problem

Sandra Gutiérrez
Departamento de Matemática
Escuela Politécnica Nacional
Quito, Ecuador
sandra.gutierrez@epn.edu.ec

Fernanda Salazar
Departamento de Matemática
Escuela Politécnica Nacional
Quito, Ecuador
fernanda.salazar@epn.edu.ec

Luis M. Torres
Centro de Modelización
Matemática (MODEMAT)
Escuela Politécnica Nacional
Quito, Ecuador
luis.torres@epn.edu.ec

Ramiro Torres
Departamento de Matemática
Escuela Politécnica Nacional
Quito, Ecuador
ramiro.torres@epn.edu.ec

Fernando Jiménez
Centro de Modelización
Matemática (MODEMAT)
Escuela Politécnica Nacional
Quito, Ecuador
fernando.jimenez@epn.edu.ec

Emilio Pérez
Facultad de Ciencias
Escuela Politécnica Nacional
Quito, Ecuador
jorge.perez@epn.edu.ec

ABSTRACT

The problem studied in this paper is motivated by the operations at the Internal Revenue Service of Ecuador. In this system, officials in charge must decide how many service counters must be active at any time to serve the incoming flow of customers and minimize the total number of man-hours that are necessary to serve such counters. This problem can be modeled as a multi-period non-preemptive parallel machine scheduling problem. In this research, two different integer programming formulations are proposed, where some lower bounds and valid inequalities are provided for both formulations. Moreover, computational results on real-world instances are reported.

KEYWORDS

Integer Programming, parallel machine scheduling, network flows, real-world instances

1 INTRODUCTION

The Internal Revenue Service of Ecuador (*Servicio de Rentas Internas, SRI*) has adopted the strategic goal of improving the quality of customer service in its agencies. One fundamental aspect in this regard is the minimization of the waiting and transaction times that citizens have to spend when carrying out their tax declaration procedures and other fiscal duties. At the same time, SRI is interested in making an optimal use of the human resources, as employees are qualified for performing customer service tasks alongside other internal duties during a day. Service counters at an agency can dynamically open or close along a work-day to properly address the influx of customers. To keep the planning simple, management officials have decided to split the work-day in one-hour time periods. Each counter can either be open or closed during a specific time period. Thus, management officials at the agencies are confronted every day with the online optimization problem of determining how many counters must be kept open at each period to serve the incoming customers. The objective is to minimize the total counter-hours required (i.e., the sum of the number of open counters over all periods), while keeping a prescribed limit on the maximum waiting time allowed for a customer.

In this work, an offline version of this problem is considered, which can be modeled as a machine scheduling problem. We are given a set of customers (jobs) and a set of service counters (machines) that must process the customers within a given time horizon, which is split into time periods. Each job is characterized by an arrival time, a latest processing start time, and a processing time. All machines are identical and each machine can either be active or idle at each period. The goal is to decide which machines must be active at which periods and to schedule all jobs to available active machines, such that the number of machine-periods (i.e., the sum of the number of active periods over all machines) is minimized, while the processing of each job starts within its deadline. Once started, the processing of a job must not be interrupted.

Scheduling problems have been studied intensively for more than 50 years by researchers in management, industrial engineering, operations research and computer science. As stated in [4], these problems are concerned with the allocation of scarce resources to activities with the objective of optimizing one ore more performance measures. The application areas range widely, from Nurse Rostering Problems, to University Timetabling Problems, to Scheduling Problems in the Airline Industry, among others. In [2] the authors consider the problem of non-preemptively Scheduling jobs with Release dates and Deadlines on a Minimum number of (identical) machines (SRDM). When all jobs have equal release times and equal deadlines, the problem is solved as a classic bin packing problem, whereas when the jobs are allowed to have slacks at most one, the problem can be solved efficiently via a network flow formulation. The slack of a job is defined as the difference between its release time and the latest possible time it may be started while still meeting its deadline. Several variants of *SRDM*, have been described in the literature. For instance, in [7] a 2-approximation algorithm and a 6-approximation algorithm are proposed for the specific cases when all jobs have a common release time and when all jobs require the same processing time, respectively. A different variant consists in considering limited workload capacity for the machines [6]. The authors propose several heuristics adapted from classic bin packing heuristics and an exact method based on a

10.48786/alioeuro.2022.10

branch-and-price approach. They also classify the corresponding scheduling problem with interval constraints into two subclasses: (i) a discrete version in which sets of possible processing intervals for the jobs are given explicitly, and (ii) a continuous version in which each job must be processed within a time window defined by its deadline and its release date. In [1], a compilation of mathematical programming formulations for a large number of scheduling problems is provided. It is shown how a network flow formulation approach can be employed for the uniform and unrelated machine makespan scheduling problem and for the scheduling problem with preemption, release times, and due dates. A recent study [5] deals with a customer service scheduling problem, where stochastic customer arrival times and service durations are included into an integer programming problem that minimizes an objective consisting of two components: (i) a penalty for customer waiting times and (ii) the cost of providing service representatives. To solve this problem efficiently, an adapted water wave optimization algorithm is used.

The problem addressed in this work can be considered as a multi-period version of SRDM. As stated above, a set of jobs is given, each of them characterized by a release time, a processing time, and a deadline. As in SRDM, all jobs must be processed non-preemptively in parallel identical machines. Additionally, the time horizon is divided into time periods and a machine can be configured to be either active or inactive at a time period. Jobs may only be scheduled to active machines. The goal is to determine the minimum number of total machine-periods (i.e., the sum of the number of active periods over all machines) required to process all jobs.

This paper is organized as follows. In the next section, two integer programming formulations are provided for the non-preemptive multi-period scheduling problem with parallel machines described above. The first one is a continuous version in which processing of a job is allowed to start within a time interval, whereas the latter corresponds to a network flow formulation considering a discretization of the aforementioned interval. Some lower bounds and valid inequalities are presented for both formulations. In Section 3 results of computational experiments on real-world instances are discussed for both formulations, revealing that the discretized formulation leads to a better performance.

2 NOTATION AND INTEGER PROGRAMMING FORMULATIONS

Let $J = \{1, \dots, n\}$ be a set of jobs and M be a set of machines. Each job $i \in J$ is associated with an arrival time $a_i \in \mathbb{R}_+$ and a processing time $t_i > 0$. Processing of each job must start at most $\gamma \in \mathbb{Z}_+$ time units after its arrival. The latest time when processing of job i is allowed to start is denoted by $b_i := a_i + \gamma$. The time horizon is divided into a (finite) set P of time periods of equal length L. A machine is allowed to be either active or idle at each period and may process a job only while it is active. Moreover, an active machine can process at most one job at the same time, and there is no limit on the number of jobs that a machine can process.

The problem consists in deciding which machines must be active at each period and scheduling all jobs on active machines, such that processing of jobs takes place within the corresponding deadlines. The objective is to minimize the total amount of machine-periods, i.e., the sum of the number of active machines over all time periods.

In the following, two mixed integer programming models are proposed for this problem. The first one is a continuous version where processing of job $i \in J$ must start within time interval $[a_i; b_i]$, while the second one uses a discretization of this interval.

2.1 First formulation

To describe the first scheduling model, a directed graph $D = (V, A)$ is defined. The set of nodes $V := J \cup \{0\}$ contains one node for each job and an additional node 0. An ordered pair of jobs $(i, j) \in J \times J$, is said to be *compatible* if job j can be performed after job i on the same machine, i.e., if $a_i + t_i \leq b_j$ holds. Let $C \subset J \times J$ denote the set of compatible jobs. The set of arcs of D is given by $A := C \cup (\{0\} \times J) \cup (J \times \{0\})$. Each directed circuit in D containing the node 0 will represent the sequence of jobs processed by a machine.

This formulation considers binary, integer and continuous variables. Let x_{ij} be a binary variable taking the value of one if the arc $(i, j) \in A$ is used in the solution, and zero otherwise. For every node $i \in J$, a continuous variable $T_i \in \mathbb{R}_+$ indicates its processing start time, while an integer variable $y_i^1 \in P$ represents the time period to which T_i belongs, and an integer variable $y_i^2 \in P$ denotes the time period containing the processing finish time $T_i + t_i$. Finally, for every pair of compatible jobs $(i, j) \in C$, the binary variable z_{ij} is equal to one if the following conditions are satisfied: job j is processed immediately after job i on the same machine, job i finishes at period q and job j starts at period $r > q$. Otherwise, z_{ij} is equal to zero.

The continuous scheduling model $(\mathcal{CSM})$ is stated as follows:

$$\min \sum_{i \in J} x_{0i} + \sum_{i \in J} (y_i^2 - y_i^1) + \sum_{(i,j) \in C} z_{ij} \tag{1}$$

$$\sum_{(i,j) \in A} x_{ij} = 1, \qquad \forall i \in J, \tag{2}$$

$$\sum_{(j,i) \in A} x_{ji} - \sum_{(i,j) \in A} x_{ij} = 0, \qquad \forall i \in J, \tag{3}$$

$$\sum_{i \in J} x_{0i} \leq |M|, \tag{4}$$

$$T_i + t_i - T_j \leq K(1 - x_{ij}), \qquad \forall (i,j) \in C, \tag{5}$$

$$L y_i^2 \geq T_i + t_i - L, \qquad \forall i \in J, \tag{6}$$

$$L y_i^1 \leq T_i, \qquad \forall i \in J, \tag{7}$$

$$|P|\, z_{ij} \geq (y_j^1 - y_i^2) - |P|\,(1 - x_{ij}), \qquad \forall (i,j) \in C, \tag{8}$$

$$a_i \leq T_i \leq b_i, \qquad \forall i \in J, \tag{9}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in A,$$

$$z_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in C,$$

$$T_i \in \mathbb{R}_+, y_i^1, y_i^2 \in P, \qquad \forall i \in J.$$

where K is a sufficiently large number.

The objective function computes the total number of machine-active periods using three terms. In the first term, one period is counted for each job that appears as a first job in a machine schedule. The second term accounts for additional periods required for jobs whose start and end of processing occurs in different time periods. Finally, the third term includes the additional periods required when a pair of compatible jobs is processed by the same machine and the second job starts at a period posterior to the end of the first job.

Constraints (2) and (3) are degree constraints on the subgraph induced by the selected arcs. Together with constrains (5) they specify that the solution must consist of a set of directed circuits, such that each circuit contains the node 0 and each node $i \in J$ is contained exactly in one circuit. Each of these circuits represents the sequence of jobs processed on a specific machine. Constraints (4) limit the number of such circuits to be less than or equal to the number of available machines. Constraints (5) set the values of the processing start times of jobs according to the job sequences assigned to each machine. Constraints (6)–(7) determine the starting and ending periods of each job. Constraints (8) count the number of times when two consecutive jobs start and end at different periods, i.e., if $(y_j^1 - y_i^2) > 0$ and $x_{ij} = 1$, then $z_{ij} = 1$. Finally, constraints (9) are time window constraints for the start time of each job.

In the following some valid inequalities for this model are presented.

Theorem 2.1. *The optimal objective value of $\mathcal{CSM}$ is not smaller than:*

$$\left\lceil \frac{\sum_{i \in J} t_i}{L} \right\rceil$$

Proof. If all machines have no idle time, then the total time for processing all jobs is equal to $\sum_{i \in J} t_i$ and the corresponding number of machine-periods is:

$$\left\lceil \frac{\sum_{i \in J} t_i}{L} \right\rceil$$

Clearly this number defines a lower bound for the optimal value of $\mathcal{CSM}$. □

The next result identifies explicitly the period in which a job is performed.

Theorem 2.2. *Given a job $i \in J$ such that $\lfloor a_i/L \rfloor = \lfloor (b_i + t_i)/L \rfloor$, Then, the equation:*

$$y_i^1 = y_i^2 = \left\lfloor \frac{a_i}{L} \right\rfloor$$

is valid for $\mathcal{CSM}$.

Proof. Since $a_i \leq T_i \leq b_i$, the result follows straightforwardly from (6)-(7). □

The time window requirements on the processing start times of jobs enforce certain binary variables to take the value zero, which in the proposed formulation directly leads to a reduction in the number of binary variables.

Theorem 2.3. *Given two jobs $i, j \in J$, such that $(i, j) \in C$ and $\lfloor a_i/L \rfloor = \lfloor b_j/L \rfloor$, then*

$$y_j^1 - y_i^2 \leq 0.$$

Moreover, the equation:

$$z_{ij} = 0$$

holds for the optimal solution of $\mathcal{CSM}$.

Proof. Since $\lfloor a_i/L \rfloor = \lfloor b_j/L \rfloor$, the processing start times of jobs i and j belong to the same time period. Thus,

$$y_j^1 \leq \left\lfloor \frac{T_j}{L} \right\rfloor \leq \left\lfloor \frac{b_j}{L} \right\rfloor = \left\lfloor \frac{a_i}{L} \right\rfloor \leq \left\lceil \frac{T_i + t_i - L}{L} \right\rceil \leq y_i^2,$$

where the last but one inequality follows from the fact that $a_i < T_i + t_i$. Moreover, since the right-hand side of (8) is less than or equal to zero, $z_{ij} = 0$ holds for an optimal solution. □

Some combinations of arcs in D represent infeasible solutions. Therefore, those subtours must be eliminated as a way to strengthen the model.

Theorem 2.4. *Given two jobs $i, j \in J$, such that $(i, j), (j, i) \in C$, the inequalities*

$$z_{ij} + z_{ji} \leq 1, \quad x_{ij} + x_{ji} \leq 1$$

are valid for $\mathcal{CSM}$.

Proof. It follows from (5) that x_{ij} and x_{ji} cannot both be equal to one at the same time. Thus, z_{ij} and z_{ji} cannot both be equal to one, neither. □

2.2 Second formulation

A discretized version of the scheduling problem, in which the sets of time intervals for job processing are explicitly given, is considered at next as an alternative to the previous formulation. We use the term *event* to refer to the start or end of the processing of a job, as well as to the start or end of a time period. For each job $i \in J$, the set Q_i^1 contains all (discrete) time points related to events concerning the processing of i, i.e., all possible times $a_i, a_i + 1, \ldots, b_i$ at which processing of i is allowed to start, together with all possible times $a_i + t_i, a_i + t_i + 1, \ldots, b_i + t_i$ at which the processing of the job may end. Moreover, $Q^2 := \{0, L, \ldots, |P|\, L\}$ is the set of time period limits, i.e., the set of times at which time periods may start or end. Finally, let $Q = \bigcup_{i \in J} Q_i^1 \cup Q^2$. In the following we assume that all events occur at different times, i.e., that $Q_i^1 \cap Q_r^1 = \emptyset$ holds for all $i, r \in J$ and $Q_i^1 \cap Q^2 = \emptyset$ holds for all $i \in J$. If two or more events coincide, one can slightly perturb the values of a_i, b_i, and t_i without changing the optimum solution of the problem instance until this assumption holds. Thus, $|Q| = 2(\gamma + 1)\,|J| + |P| + 1$. The discretized scheduling problem can then be formulated as an instance of the minimum cost flow problem as follows.

Let $D = (V, A)$ be a digraph having one node for each event and two additional nodes representing the start and end of the schedule, i.e., $V = \{1, \ldots, |Q|\} \cup \{o, d\}$. Moreover, let h be a function assigning to each node $j \in \{1, \ldots, |Q|\}$ the time $h(j) \in Q$ of the event represented by j and assume the nodes have been labeled in such a way that $j < l$ holds if and only if $h(j) < h(l)$.

The set of arcs of D is the union of three subsets A_1, A_2, A_3, where:

$$A_1 = \{(j, j+1) : 1 \leq j < |Q|\} \cup \{(o, 1), (|Q|, d), (o, d)\}$$
$$A_2 = \{(j, l) \,:\, h(j), h(l) \in Q^2, h(l) = h(j) + L\}$$
$$A_3 = \cup_{i \in J} A_3^i, \text{ with } A_3^i = \{(j, l) \,:\, h(j), h(l) \in Q_i^1,\ h(l) = h(j) + t_i\}.$$

Observe that $|A_3| = (\gamma + 1)\,|J|$.

The arc cost function $c : A \longrightarrow \mathbb{R}_+$ is defined in the following way. All arcs in A_1 of the form $(j, j+1)$, where $h(j) \in Q^2$ have costs equal to one, the remaining arcs in A_1 have costs equal to zero. All arcs in A_2 have costs equal to zero, whereas the cost of each arc $(j, l) \in A_3$ is set to $\left\lfloor \frac{h(l)}{L} \right\rfloor - \left\lfloor \frac{h(j)}{L} \right\rfloor$.

Node demands are given by

$$g_j = \begin{cases} -\,|M|\,, & \text{if } j = o, \\ |M|\,, & \text{if } j = d, \\ 0, & \text{si } j \in V \setminus \{o, d\}. \end{cases}$$

Finally, the capacity u_{jl} of an arc (j, l) is defined to be equal to $|M|$, if $(j, l) \in A_1 \cup A_2$, and equal to one, if $(j, l) \in A_3$.

Example 2.5. Consider a time horizon consisting of two time periods, each one having length $L = 7$ time units . Two jobs have to be scheduled within this horizon on three parallel machines. The first job arrives at time $a_1 = 1$ and requires a processing time of $t_1 = 5$ time units, while the arrival time of the second job is $a_2 = 4$ and its processing time is $t_2 = 6$ time units. The maximum waiting time for processing start is $\gamma = 1$ time unit. Thus, the first job has to be processed in one of the intervals $[1; 6]$ or $[2; 7]$, while processing of the second job must occur in one of the intervals $[4; 10]$ or $[5; 11]$. Therefore, $Q_1^1 = \{1, 2, 6, 7\}$, $Q_2^1 = \{4, 5, 10, 11\}$, and $Q^2 = \{0, 7, 14\}$. Figure 1 depicts the graph construction for this instance. Nodes with dashed border belong to Q^2 and the remaining nodes belong to Q^1. Observe that $h(7) = h(8) = 7$, as the end of the second interval for job 1 coincides with the end of the first time period. Nonetheless, these two different events are represented by two different nodes to achieve the number of $2(\gamma + 1)\,|J| + |P| + 3$ nodes proposed in the second formulation.

One feasible solution consists in processing each job at the first possible time on a different machine. The cost of the solution is three, as the machine processing the first job is active during the first period, whereas the machine processing the second job needs to be active during the two periods in the time horizon. This solution is represented in the graph as three (o, d)-paths P_1, P_2, and P_3, consisting of the nodes:

$$P_1 : o, 1, 2, 6, 7, 8, 11, d.$$
$$P_2 : o, 1, 2, 3, 4, 9, 10, 11, d.$$
$$P_3 : o, d.$$

Paths P_1 and P_2 indicate the job sequence processed by the first two machines, while path P_3 signalizes that the third machine has no job assigned in the solution. Costs of the paths correspond to the number of active periods for the machines: 1, 2, and 0 for the three given paths.

The flow based formulation ($\mathcal{DSM}$) can be stated as follows:

$$\min \sum_{(j,l)\in A} c_{jl} x_{jl} \tag{10}$$
$$\sum_{(j,l)\in A_3^i} x_{jl} = 1, \qquad \forall i \in J, \tag{11}$$
$$\sum_{(l,j)\in A} x_{lj} - \sum_{(j,l)\in A} x_{jl} = g_j, \qquad \forall j \in V, \tag{12}$$
$$x_{jl} \le u_{jl}, \qquad \forall (j,l) \in A, \tag{13}$$
$$x_{jl} \in \mathbb{Z}_+, \qquad \forall (j,l) \in A.$$

As observed in Example 2.5, any feasible solution to $\mathcal{DSM}$ can be decomposed into at most $|M|$ (o, d)-paths in D. For any such path $\hat{P}$, the arcs in $A(\hat{P}) \cap A_3$ determine the sequence of jobs to be processed by one of the machines. Thus, the objective function seeks the minimization of the total number of machine-periods. Constraints (11) ensure that every job is processed at one of its given feasible intervals. Constraints (12) and (13) are flow demand and arc capacity constraints, respectively. Together, they require that at most $|M|$ machines are used.

As in the case of $\mathcal{CSM}$, some valid inequalities for $\mathcal{DSM}$ are stated and proven at next. Firstly, since the objective function measures the total number of machine-periods used in the solution, the next result is obtained with the same arguments from Theorem 2.1:

Theorem 2.6. *The inequality*

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \ge \left\lceil \frac{\sum_{i\in J} t_i}{L} \right\rceil$$

is valid for $\mathcal{DSM}$.

For any period $p \in P$, let $J_p = \{i \in J : L(p-1) \le a_i < b_i + t_i \le Lp\}$ be the set of jobs whose processing must start and end within p and let $V_p = \{j \in V \setminus \{o, d\} : L(p-1) \le h(j) \le Lp\}$ be the set of nodes representing events that occur within p. Comparing the number of active machines in p with the total processing time of jobs in J_p, the following result is obtained.

Theorem 2.7. *For every period $p \in P$, the inequality*

$$\sum_{\substack{(j,l)\in A_3: \\ j \notin V_p, l \in V_p}} x_{jl} + x_{k,k+1} \ge \left\lceil \frac{\sum_{i\in J_p} t_i}{L} \right\rceil$$

is valid for $\mathcal{DSM}$. Here, k is the node representing the starting time of period p, i.e., $h(k) = L(p-1) \in Q^2$.

Proof. Let $p \in P$. The expression in the left-hand side equals to the number of active machines in period p. At the time $L(p-1)$ when the period starts, each of these machines is either busy processing a job or idle and waiting for the next job. The sum at the beginning of the expression accounts for set of busy machines while the number of idle active machines is given by $x_{k,k+1}$.

Since the total time required for processing all jobs that have to start in p is equal to $\sum_{i\in J_p} t_i$, the right-hand side of the inequality provides a lower bound on the number of required active machines. □

Theorem 2.8. *The inequality*

$$x_{o1} \ge \max_{p\in P} \left\{ \left\lceil \frac{\sum_{i\in J_p} t_i}{L} \right\rceil \right\}$$

is valid for $\mathcal{DSM}$.

Proof. Clearly, from the arguments in the proof of the previous theorem, the right-hand side of the inequality is a lower bound for the minimum number of machines required in any feasible solution.

On the other hand, the schedule of each machine used in the solution is represented by an (o, d)-path different from the arc (o, d). Since each of these paths contains the arc $(o, 1)$, the number of such paths equals to x_{o1}. □

3 COMPUTATIONAL RESULTS

In this section results of some computational experiments carried out with both IP formulations are reported. A set of tests instances has been constructed in order to assess the performance of both formulations, $\mathcal{CSM}$ and $\mathcal{DSM}$, considering the following two settings: with and without the inclusion of the valid inequalities described in Section 2. The test instances are samples of real-world (anonymized) data provided by the SRI from its internal service database. Data include customers arrival times, waiting times and processing times for some selected work-days in October 2017 and June 2018 at one of the largest agencies in Ecuador. All instances were obtained by sampling a partial number of customers from these days.

The IPs were solved using the integer programming solver Gurobi 9.1.1 [3] in its default configuration and the Python programming language interface. All experiments were performed

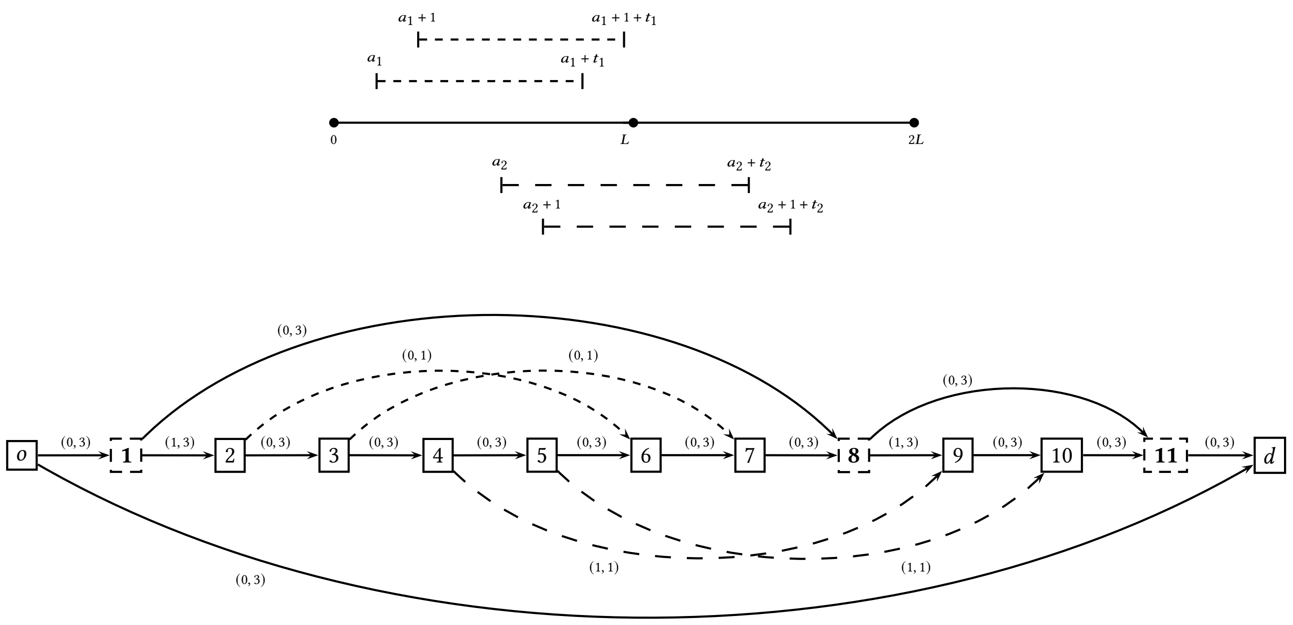

Figure 1: Graph construction for $\mathcal{DSM}$. Labels (c_{ij}, u_{ij}) on the arcs indicate arc costs and capacities, respectively.

on a laptop Intel Core i7 2.7GHz with 16 GB RAM running Windows 10 Home. The computation time was limited to 3600 seconds for each instance.

The objective of the experiments was to compare the performance of both IP formulations on the aforementioned instances, as well as to measure the effect of strenghtening the formulations with the valid inequalities presented in the previous section. Table 1 and Table 2 summarize the results obtained using Gurobi in its default configuration and including lower bounds and valid inequalities, respectively. There, the first two columns display the number of jobs and the number of periods in each instance; columns 3 to 9 report data for the $\mathcal{CSM}$ formulation, including the best objective function value, the best lower bound, the relative optimality gap, the CPU time in seconds, the number of B&B nodes explored, the number of variables, and the number of constraints; the remaining columns are dedicated to the $\mathcal{DSM}$ formulation showing the objective function value, the relative optimality gap, the CPU time in seconds, the number of explored B&B nodes, the number of variables, and the number of constraints. All instances consider 25 identical machines.

Regarding to the size of the models, observe that $\mathcal{CSM}$ is clearly larger having up to 17.6 times more variables and 38.6 times more constraints than $\mathcal{DSM}$. This can be explained because in the first formulation both the number of variables and the number of constraints depend on the set of compatible jobs, which has size of order $O(|J|^2)$. On the other hand, these numbers depend linearly on the number of jobs, the number of periods and the maximum allowed waiting times in $\mathcal{DSM}$.

Formulation $\mathcal{DSM}$ performed clearly better than $\mathcal{CSM}$ in both settings (with or without the inclusion of lower bounds and valid inequalities), as it is shown in Table 1 and Table 2. In the first setting, $\mathcal{DSM}$ solves all instances to optimality, five of them in less than a minute, whereas $\mathcal{CSM}$ fails to solve any instance within the maximum allowed running time. Even more, the minimum reported gap is 50% and no feasible solution is found for the instance with the largest number of jobs.

When lower bounds and valid inequalities are included in $\mathcal{DSM}$ all instances are still solved to optimality, but the corresponding running times change: they decrease for five of the eight instances and increase for the other three. On the other hand, no optimal solution is found for any of the test instances in model $\mathcal{CSM}$ under this setting. However, the inclusion of the valid inequalities is clearly relevant since tighter lower bounds are obtained for all instances, reaching an average gap equal to 16.15%. Moreover, feasible solutions are found for all instances in this setting.

REFERENCES

[1] Jacek Blazewicz, Moshe Dror, and Jan Weglarz. 1991. Mathematical programming formulations for machine scheduling: A survey. *European Journal of Operational Research* 51, 3 (April 1991), 283–300. https://doi.org/10.1016/0377-2217(91)90304-e

[2] Mark Cieliebak, Thomas Erlebach, Fabian Hennecke, Birgitta Weber, and Peter Widmayer. [n.d.]. Scheduling With Release Times and Deadlines on A Minimum Number of Machines. In *IFIP International Federation for Information Processing*. Kluwer Academic Publishers, 209–222. https://doi.org/10.1007/1-4020-8141-3_18

[3] Inc. Gurobi Optimization. 2018. Gurobi Optimizer Reference Manual. http://www.gurobi.com

[4] Joseph YT Leung. 2004. *Handbook of scheduling: algorithms, models, and performance analysis.* CRC press.

[5] Xueqin Lu, Chenxin Wu, Xuhua Yang, Minxia Zhang, and Yujun Zheng. 2021. Adapted water wave optimization for integrated bank customer service representative scheduling. *International Journal of Production Research* (June 2021), 1–16. https://doi.org/10.1080/00207543.2021.1942284

[6] Luis Osorio-Valenzuela, Jordi Pereira, Franco Quezada, and Óscar C. Vásquez. 2019. Minimizing the number of machines with limited workload capacity for scheduling jobs with interval constraints. *Applied Mathematical Modelling* 74 (Oct. 2019), 512–527. https://doi.org/10.1016/j.apm.2019.05.007

[7] Guosong Yu and Guochuan Zhang. 2009. Scheduling with a minimum number of machines. *Operations Research Letters* 37, 2 (March 2009), 97–101. https://doi.org/10.1016/j.orl.2009.01.008

Table 1: Solving $\mathcal{CSM}$ and $\mathcal{DSM}$ using Gurobi in default configuration.

		$\mathcal{CSM}$							$\mathcal{DSM}$					
$\|J\|$	$\|P\|$	Obj	L. B.	Gap(%)	Time(s)	# Nodes	# Vars	# Constr	Obj	Gap(%)	Time(s)	# Nodes	# Vars	# Constr
30	1	7	1	85.71	3600.07	221175	1572	1603	7	0.0	0.1	0	1235	634
50	9	12	6	50.0	3600.07	251679	2920	2971	12	0.0	1.04	0	2535	1490
248	9	43	4	90.7	3600.07	3457	65496	65745	39	0.0	397.06	14645	10595	5580
287	4	51	5	90.2	3600.08	2456	91639	91927	46	0.0	28.05	261	11833	6072
372	9	69	7	89.85	3600.92	533	146374	146747	60	0.0	627.06	4582	16102	8607
457	5	90	7	92.22	3606.06	3	227165	227623	74	0.0	44.12	397	18404	9243
500	6	96	3	96.88	3600.17	0	275122	275623	73	0.0	23.92	424	18416	8388
600	7	inf	4	inf	3627.82	0	389886	390487	88	0.0	502.28	2990	22151	10115

Table 2: Solving $\mathcal{CSM}$ and $\mathcal{DSM}$ using lower bounds and valid inequalities

		$\mathcal{CSM}$							$\mathcal{DSM}$					
$\|J\|$	$\|P\|$	Obj	L. B.	Gap(%)	Time(s)	# Nodes	# Vars	# Constr	Obj	Gap(%)	Time(s)	# Nodes	# Vars	# Constr
30	1	7	6	14.28	3600.07	1402599	1572	2134	7	0.0	0.06	0	1235	637
50	9	12	8	33.33	3600.18	453175	3145	2920	12	0.0	0.58	0	2535	1501
248	9	42	37	11.90	3600.11	3164	65496	68532	39	0.0	366.13	22588	10595	5592
287	4	48	44	8.33	3600.06	3403	91639	99903	46	0.0	26.45	461	11833	6074
372	9	63	56	11.11	3600.14	2498	146374	152987	60	0.0	1435.19	21197	16102	8609
457	5	79	68	13.92	3600.18	1967	227165	243898	74	0.0	61.21	701	18404	9245
500	6	82	67	18.29	3600.30	1601	275122	275624	73	0.0	28.54	0	18416	8397
600	7	100	82	18.00	3600.47	875	389866	390488	88	0.0	93.33	782	22151	10117