Time Representations and Mathematical Models for Process Scheduling Problems

Sylvain Mouret[a], Ignacio E. Grossmann[a,*], Pierre Pestiaux[b]

[a]*Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213, USA*
[b]*Total Refining & Marketing, Research Division, 76700 Harfleur, France*

Abstract

During the last 15 years, many mathematical models have been developed in order to solve process operation scheduling problems, using discrete or continuous time representations. In this paper, we present a unified representation and modeling approach for process scheduling problems. Four different time representations are presented, compared, and applied to single-stage and multi-stage batch scheduling problems, as well as crude-oil operations scheduling problems. We introduce three solution methods that can be used to achieve global optimality or obtain near-optimal solutions depending on the stop criterion used. Computational results show that the Multi-Operation Sequencing time representation is superior to the others as it allows efficient symmetry-breaking and requires fewer priority-slots, thus leading to smaller model sizes.

Keywords:
time representations, batch scheduling, crude-oil scheduling, mixed-integer linear programming

1. Introduction

Rigorous optimization of real-world problems are often based on advanced programming tools such as mixed-integer linear programming (MILP) or constraint programming (CP, see Rossi et al. (2006)). These tools rely on a mathematical or symbolic representation of the problem which is applied by an end-user. In some cases, the relationship between the problem description and its mathematical model is not clear. Therefore an intermediate step is included in the optimization approach (see Figure 1). In this step, the representation used is detailed and approximations are made. For instance, in the context of scheduling problems, using a discrete-time formulation is in general a constraining approximation of the actual problem, and thus, it may lead to a suboptimal solution as discussed in Floudas and Lin (2004).

Additionally, it is important to note that several mathematical models may be used to obtain the global optimal solution of the problem, which is the best possible solution according to a given optimization criterion. For example, many continuous-time representations rely on a specific parameter representing the number of time points (Kondili et al., 1993), time intervals (Lee et al., 1996), or event points (Ierapetritou and Floudas, 1998) used. Therefore, the scheduling problem is represented by an infinite set of mathematical models, one for each possible value of this parameter (all positive integers). The global optimal schedule is the best solution among the optimal solution of all these models. In general, it is not possible to know a priori the parameter value that will lead to the global optimal solution, although it is sometimes possible to derive upper and lower bounds for it. The common trade-off is that global optimality may be guaranteed with a large value of this parameter which often results in prohibitive solution times.

Many different time representations have been introduced to solve scheduling problems (for review see Floudas and Lin (2004)). Experience has shown that, depending on the characteristics of the problem, some time representations are better suited than others. In this paper, we focus on scheduling problems which rely on:

[*]Corresponding author

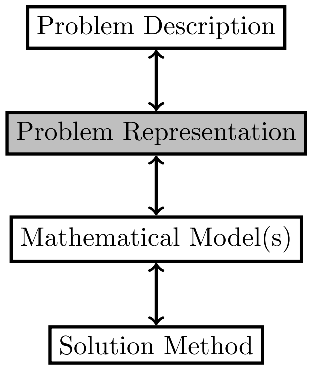


Figure 1: Four steps optimization method.

a) a set of possible *operations*, or actions, that can be performed once, several times, or not at all;

b) *scheduling decisions* that involve both selecting, parametrizing and sequencing the operations that should be executed;

c) *scheduling constraints* such as release dates, due dates, bounds on processing times, non-overlapping constraint, cardinality constraints, and precedence constraints;

d) additional *side constraints* that are used to model more complex features such as limited inventory management or process constraints.

It should be noted that, for example, the selection of operations may correspond to the selection of equipment or discrete resources for tasks in a state-task-network (Kondili et al., 1993) or in a resource-task-network (Pantelides, 1994). In general, operations are defined by fully disaggregating all possible discrete selections of actions in the scheduling system. In contrast, parameterization of operations corresponds to continuous decisions such as batch sizes, transfer volumes, or process operating conditions.

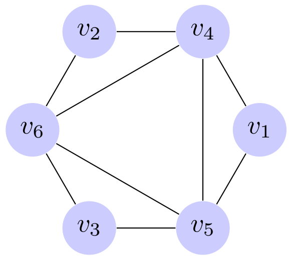
The main objective of this work is to develop a unified modeling approach for scheduling problems in order to facilitate the evaluation of several time representations, both in terms of computational time and solution quality. First, a simple scheduling problem is introduced as an example. Next, we study four different types of time representations, which have been used in the literature and clarify the relationships between them. Then, basic MILP models for pure scheduling constraints are presented for each of these time representation. Using concepts from graph theory (cliques and bicliques), we show how these models can be generally strengthened based on the structure of the scheduling problem. Two solution methods are then developed to solve these mathematical formulations. Finally, three types of problems are presented and solved using the different approaches in order to show the effectiveness of the strengthened formulations and to provide elements of comparison between the different time representations.

2. Case-Study

We introduce a small scheduling system that involves 6 different operations $v_1, \ldots, v_6$ and 3 unary resources r_1, r_2, r_3. A unary resource cannot be shared by two or more processing operations at a given time. Table 1 displays resource requirement for each operation. In this case and in the examples studied in this paper, unary resource requirements are handled as non-overlapping constraints between operations. For instance, operations v_1 and v_4 cannot overlap as they both use resource r_1. Also, operations v_5 and v_6 cannot overlap as they both use resource r_3. Besides, as a given operation v can be executed several times, any two separate executions of v may not overlap. Thus, any operation v cannot overlap with itself. Different

Table 1: Case-study: resource requirements

Operation	v_1	v_2	v_3	v_4	v_5	v_6
Resources	r_1	r_2	r_3	$r_1 \wedge r_2$	$r_1 \wedge r_3$	$r_2 \wedge r_3$



$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 2: Non-overlapping matrix and graph for case-study.

linear objectives can be considered: maximization of profit, minimization of makespan, minimization of assignment costs, minimization of tardiness or earliness.

In order to extract useful information from the structure of the problem, we use a global representation of all the non-overlapping constraints. The *non-overlapping matrix*, denoted by NO, is such that $NO_{vv'} = 1$ if operation v and v' must not overlap, 0 otherwise. The *non-overlapping graph*, denoted by $G_{NO} = (V, E)$, is an undirected graph where the set of vertices v is the set of operations and the set of edges is defined by $E = \{\{v, v'\} \text{ s.t. } NO_{vv'} = 1\}$. Therefore, the non-overlapping matrix is the adjacency matrix of graph G_{NO}. The concept of non-overlapping graph can be viewed as an extension of the disjunctive graph (Adams et al., 1988), which is used to represent disjunctive constraints between operations that have to be executed exactly once. In this paper, we consider operations that can be executed once, several times, or not at all. Figure 2 shows the non-overlapping matrix and graph for the case-study. For clarity, edges that connect a vertex to itself, called self-loops, are not represented.

3. Time Representations

In this paper, we study four different time representations and show how they can be defined using identical concepts. Each of these make use of priority-slots, which are used to assign and order the executions of operations. The number of priority-slots has to be postulated a priori, and each priority-slot corresponds to a position in a sequence. Whenever an operation is assigned to a priority-slot, it has to be executed with a corresponding scheduling priority. Any operation may be executed several times by assigning it to multiple priority-slots. It is not straightforward to select the best number of priority-slots. Indeed, postulating a large number priority-slots increases the chance of obtaining the global optimal solution, but it also increases the size of the model and the CPU time. The four time representations are listed below.

a. Multi Operation Sequencing (MOS)

b. Multi Operation Sequencing with Synchronized Start Times (MOS-SST)

c. Multi Operation Sequencing with Fixed Start Times (MOS-FST)

d. Single Operation Sequencing (SOS)

Figure 3 shows how the same schedule for the case-study can be obtained within each time representations. Each execution of an operation is represented by an horizontal bar in the upper Gantt chart, while resource usage is represented by horizontal lines in the lower Gantt chart. The priority-slots are represented by number labels on each operation execution. In each case,

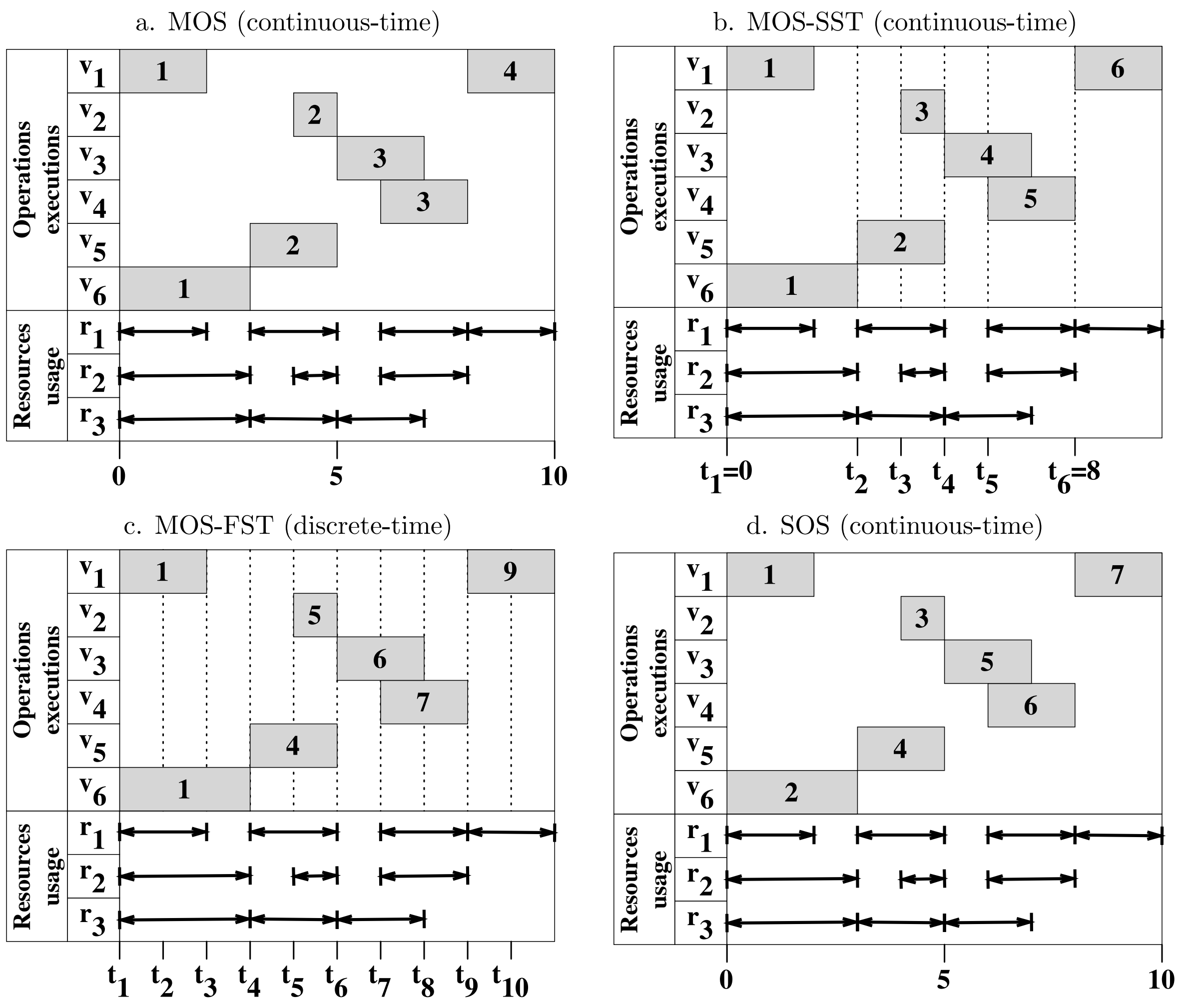


Figure 3: A unique schedule obtained through different time representations.

the smallest possible number of priority-slots needed to obtained the solution has been used. From this figure, it is clear that some time representations require more priority-slots than others.

In the **MOS** representation, several operations can be assigned to each priority-slots as long as they may overlap with each other. For instance, in Figure 3(a) operations v_1 and v_6 are allowed to overlap and are both assigned to the first priority-slot. However, operations v_1 and v_2 cannot overlap and are consequently assigned to different priority-slots: slots 1 and 4 for operation v_1, slot 2 for operation v_2. If two non-overlapping operations v and v' are assigned to priority-slots i and j, respectively, such that $i < j$, then operation v' must be executed after operation v. For instance, operation v_1 assigned to priority-slot 4 is executed after operation v_2 assigned to priority-slot 2. We denote $\mathbf{MOS}(n)$ a scheduling model using the MOS time representation with n postulated priority-slots. This time representation was introduced by Ierapetritou and Floudas (1998) as the event point formulation. Their mathematical model, although significantly different than the model developed in this paper, was used to solve several STN problems. As mentioned by Maravelias and Grossmann (2003), inventory tracking using event points is quite different than inventory tracking using time points, which might lead to inconsistent enforcement of storage capacity constraints. This issue was addressed by Janak et al. (2004) by adding additional storage tasks in the STN problem, which can lead to a significant increase of model size.

The **MOS-SST** representation is based on the same features as the MOS representation. Additionally, all operations assigned to the same priority-slot must have the same start time. For instance, in Figure 3(b), operations v_1 and v_6 are both assigned to priority-slot 1, and therefore both start at the same time $t = 0$. Thus, each priority-slot i is associated to variable time-point t_i which is represented by a vertical dotted line in Figure 3(b). The time interval between any two successive time-points is variable. We denote **MOS-SST**(n) a scheduling model using the MOS-SST time representation with n postulated priority-slots. This type of representation has been used to solve a wide variety of problems where time-points are used to track both the start and end events of each operation (see Zhang and Sargent, 1996; Schilling and Pantelides, 1996; Maravelias and Grossmann, 2003).

The **MOS-FST** representation is based on the same features as the MOS-SST representation. Additionally, the time-point associated to each priority-slot is fixed a priori. Thus, the interval between any two successive time-points is fixed. For instance, the solution depicted in Figure 3(c) is obtained using time-points that are uniformly spaced along the time horizon: $t_1 = 0, t_2 = 1, \ldots, t_{10} = 9$. Therefore, operation v_5 assigned to priority-slot 4 starts at $t = t_4 = 3$ while operation v_4 assigned to priority-slot 7 starts at $t = t_7 = 6$. We denote **MOS-FST**(n) a scheduling model using the MOS-FST time representation with n postulated priority-slots. Discrete-time formulation for process scheduling problems were initially developed to solve STN and RTN models where processing times are assumed to be constant (see Kondili et al., 1993; Pantelides, 1994).

In the **SOS** representation, at most one operation can be assigned to each priority-slot. It is therefore possible to represent the scheduling solution by a sequence of operations (Mouret et al., 2009). For instance, the solution depicted in Figure 3(d) is represented by sequence of operations **1625341**. Similarly to the MOS model, if two non-overlapping operations v and v' are assigned to priority-slots i and j $(i < j)$, then v' must be executed after v. We denote **SOS**(n) a scheduling model using the SOS time representation with n postulated priority-slots. This time representation was introduced by Mouret et al. (2009) to solve the refinery crude-oil operations scheduling problem.

From these definitions, it can be inferred that for a given number of priority-slots n the integer feasible space of **MOS**(n) is larger than the integer feasible space of models **MOS-SST**(n), **MOS-FST**(n), and **SOS**(n). Indeed, the latter models are derived from the MOS model by introducing additional constraints, which reduce the set of feasible solutions. Furthermore, the integer feasible space of **MOS-SST**(n) is larger than the one of **MOS-FST**(n) and **SOS**(n). In particular, any solution for the **SOS** model is a solution for the **MOS-SST** model. Indeed, at most one operation can be assigned to each priority-slot so the synchronization of start times is automatically satisfied.

These properties can also be interpreted by considering a scheduling solution z. We denote $z \in \mathbf{MOS}(n)$ the membership of schedule z to the integer feasible space of model **MOS**(n). In other words, $z \in \mathbf{MOS}(n)$ means that solution z satisfies all the constraints of model **MOS**(n). We introduce the minimum number of priority-slots needed to "find" solution z using each time representation.

$$\begin{aligned}
n_{\mathbf{MOS}}(z) &= \min_n \{n | z \in \mathbf{MOS}(n)\}\\
n_{\mathbf{MOS\text{-}SST}}(z) &= \min_n \{n | z \in \mathbf{MOS\text{-}SST}(n)\}\\
n_{\mathbf{MOS\text{-}FST}}(z) &= \min_n \{n | z \in \mathbf{MOS\text{-}FST}(n)\}\\
n_{\mathbf{SOS}}(z) &= \min_n \{n | z \in \mathbf{SOS}(n)\}
\end{aligned}$$

Then, the following inequalities hold:

$$\begin{cases}
n_{\mathbf{MOS}}(z) \leq n_{\mathbf{MOS\text{-}SST}}(z) \leq n_{\mathbf{MOS\text{-}FST}}(z)\\
n_{\mathbf{MOS}}(z) \leq n_{\mathbf{MOS\text{-}SST}}(z) \leq n_{\mathbf{SOS}}(z)
\end{cases}$$

Remark. An important limitation of these time representations is that operations are considered as a whole for sequencing purpose. More precisely, a scheduling priority is assigned to operations and not to the start and end events of these operations, which may be necessary to solve some scheduling problems. For instance, operations that require a cumulative resource with capacity greater than 1 (e.g. manpower limited to 2 workers) are allowed to overlap, but only a limited number of these operations may overlap at any given point in time. The models developed in this paper do not accommodate these features. Another case is inventory tracking when simultaneous charging and discharging of tanks is allowed. It is sufficient to enforce capacity limitations only at the start and end events of each charging/discharging operations, but in order to do so, a precise sequence of such events needs to be obtained by the model. Possible workarounds or extensions of the model to handle these specific features, such as presented in Janak et al. (2004), will not be discussed in this paper.

4. Mathematical Models

In this section, we present mathematical models for each time representation. They all rely on the same sets, parameters and variables. Objective functions are not presented here (e.g. minimize makespan, minimize tardiness, or maximize profit) although they can significantly impact the solution of the corresponding formulation.

4.1. Sets and Parameters

The following sets and parameters are used.

- $T = \{1, \ldots, n\}$ is a totally ordered set of priority-slots (indices i, j, i_1, i_2).
- W is the set of all operations (indices v, v', v_1, v_2).
- H is the scheduling horizon.
- $[\underline{S_v}, \overline{S_v}] \subset [0, H]$ are bounds on the start time of any execution of operation v.
- $[\underline{D_v}, \overline{D_v}] \subset [0, H]$ are bounds on the duration of any execution of operation v.
- $[\underline{E_v}, \overline{E_v}] \subset [0, H]$ are bounds on the end time of any execution of operation v.
- $[\underline{N_v}, \overline{N_v}]$ are bounds on the total number of executions of operation v.
- $[\underline{N_{W'}}, \overline{N_{W'}}]$ are bounds on the total number of executions of all operations in W'.
- $NO_{v_1 v_2}$ is 1 if operations v_1 and v_2 must not overlap, 0 if they are allowed to overlap.
- $TR_{v_1 v_2}$ is a sequence-dependent transition time between non-overlapping operations v_1 and v_2.
- $TR_{W'}$ is a unique set transition time between any pair of non-overlapping operations in W'.
- $P_{v_1 v_2} = 1$ denotes a precedence constraint between operations v_1 and v_2.
- $P_{W_1 W_2} = 1$ denotes a precedence constraint between set of operations W_1 and W_2.

Remark 1: It should be noted that for operations with fixed processing time, $\underline{D_v} = \overline{D_v} = D_v$.

Remark 2: A set transition time $TR_{W'}$ is defined when $\forall v_1, v_2 \in W', TR_{v_1,v_2} = TR_{W'}$. It can be used to represent unit changeover times.

Remark 3: A precedence constraint between operations v_1 and v_2 states that v_1 must be executed before v_2. This implies that each operation must be executed exactly once ($\underline{N_{v_1}} = \underline{N_{v_2}} = \overline{N_{v_1}} = \overline{N_{v_2}} = 1$). A precedence constraint between sets of operations W_1 and W_2 states that exactly one operation in each set must be executed ($\underline{N_{W_1}} = \underline{N_{W_2}} = \overline{N_{W_1}} = \overline{N_{W_2}} = 1$) and the operation selected from W_1 must be executed before the operation selected from W_2.

4.2. Variables

The variables used in all models are composed of binary assignment variables, and continuous time variables.

- **Assignment variables** $Z_{iv} \in \{0,1\} \qquad i \in T, v \in W$

  $Z_{iv} = 1$ if operation v is assigned to priority-slot i, $Z_{iv} = 0$ otherwise.

- **Time variables** $S_{iv} \geq 0, D_{iv} \geq 0, E_{iv} \geq 0 \qquad i \in T, v \in W$

  S_{iv} is the start time of operation v if it is assigned to priority-slot i, $S_{iv} = 0$ otherwise.

  D_{iv} is the duration of operation v if it is assigned to priority-slot i, $D_{iv} = 0$ otherwise.

  E_{iv} is the end time of operation v if it is assigned to priority-slot i, $E_{iv} = 0$ otherwise.

4.3. MOS Model

Variable Bound Constraints. Bounds on time variables can be expressed using the following constraints.

$$\underline{S_v} \cdot Z_{iv} \leq S_{iv} \leq \overline{S_v} \cdot Z_{iv} \qquad i \in T, v \in W \tag{1a}$$

$$\underline{D_v} \cdot Z_{iv} \leq D_{iv} \leq \overline{D_v} \cdot Z_{iv} \qquad i \in T, v \in W \tag{1b}$$

$$\underline{E_v} \cdot Z_{iv} \leq E_{iv} \leq \overline{E_v} \cdot Z_{iv} \qquad i \in T, v \in W \tag{1c}$$

Time Constraint. Time variables are linked through the following additional constraint.

$$E_{iv} = S_{iv} + D_{iv} \qquad i \in T, v \in W \tag{2}$$

Cardinality constraint. The total number of execution of operations in a set $W' \subset W$ is restricted by the following constraint. A cardinality constraint on a single operation v can be enforced by setting $W' = \{v\}$.

$$\underline{N_{W'}} \leq \sum_{\substack{i \in T \\ v \in W'}} Z_{iv} \leq \overline{N_{W'}} \qquad W' \subset W \tag{3}$$

Assignment Constraint. Two non-overlapping operations v_1 and v_2 such that $NO_{v_1v_2} = 1$ cannot be assigned simultaneously to the same priority-slot.

$$Z_{iv_1} + Z_{iv_2} \leq 1 \qquad i \in T, v_1, v_2 \in W, NO_{v_1v_2} = 1 \tag{4}$$

Non-overlapping Constraint. A non-overlapping constraint between two operations $v_1, v_2 \in W$ states that they must not be executed simultaneously. This property is enforced using the following big-M constraints where the big-M constant is defined as a valid upper bound of the left hand side of the inequality.

$$E_{i_1v_1} \leq S_{i_2v_2} + H \cdot (1 - Z_{i_2v_2}) \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1v_2} = 1 \tag{5a}$$

$$E_{i_1v_2} \leq S_{i_2v_1} + H \cdot (1 - Z_{i_2v_1}) \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1v_2} = 1 \tag{5b}$$

Due to the assignment constraint (4) and variable bound constraints (1a) and (1c), equations (5a) and (5b) can be combined in order to form the following tighter surrogate constraint (see Appendix A).

$$E_{i_1v_1} + E_{i_1v_2} \leq S_{i_2v_1} + S_{i_2v_2} + H \cdot (1 - Z_{i_2v_1} - Z_{i_2v_2}) \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1v_2} = 1 \tag{6}$$

Non-overlapping Constraint with Transition Times. Sequence-dependent transition times $TR_{v_1v_2}$ between operations $v_1 \in W$ and $v_2 \in W$ can be enforced as follows.

$$E_{i_1v_1} + TR_{v_1v_2} \cdot Z_{i_1v_1} \leq S_{i_2v_2} + (H + TR_{v_1v_2}) \cdot (1 - Z_{i_2v_2}) \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1v_2} = 1 \tag{7a}$$

$$E_{i_1v_2} + TR_{v_2v_1} \cdot Z_{i_1v_2} \leq S_{i_2v_1} + (H + TR_{v_2v_1}) \cdot (1 - Z_{i_2v_1}) \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1v_2} = 1 \tag{7b}$$

If the transition time is not sequence-dependent (i.e. $TR_{v_1v_2} = TR_{v_2v_1}$) then constraints (7) can be combined in order to form the following tighter surrogate constraint.

$$\begin{gathered} E_{i_1v_1} + E_{i_1v_2} + TR_{v_1v_2} \cdot (Z_{i_1v_1} + Z_{i_1v_2}) \leq S_{i_2v_1} + S_{i_2v_2} + (H + TR_{v_1v_2}) \cdot (1 - Z_{i_2v_1} - Z_{i_2v_2}) \\ i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1v_2} = 1, TR_{v_1v_2} = TR_{v_2v_1} \end{gathered} \tag{8}$$

Precedence Constraint. Consider two operations $v_1, v_2 \in W$ such as $P_{v_1v_2} = 1$, then the precedence constraint can be expressed as follows. Note that in each summation, exactly one term can be non-zero due to the cardinality constraint.

$$\sum_{i \in T} E_{iv_1} \leq \sum_{i \in T} S_{iv_2} \qquad v_1, v_2 \in W, P_{v_1v_2} = 1 \tag{9}$$

Consider two sets of operations $W_1, W_2 \subset W$. If $P_{W_1W_2} = 1$, the previous constraint can be extended as follows. Note that in each summation, exactly one term can be non-zero due to the corresponding cardinality constraint (3).

$$\sum_{i \in T} \sum_{v_1 \in W_1} E_{iv_1} \leq \sum_{i \in T} \sum_{v_2 \in W_2} S_{iv_2} \qquad W_1, W_2 \subset W, P_{W_1W_2} = 1 \tag{10}$$

4.4. MOS-SST Model

In the MOS-SST model, the start times of all operations assigned to the same priority-slot i have to be synchronized. Therefore, we introduce positive synchronization time-points variables t_i ($i \in T$). Variables t_i correspond to the start time of all operations assigned to priority-slot i.

$$t_i \in [0, H] \qquad i \in T \tag{11}$$

Time-point Sequence Constraint. The following constraint insures that the variables t_i are ordered in time.

$$t_{i-1} \leq t_i \qquad i \in T \tag{12}$$

Synchronization Constraints. If operation v is assigned to priority-slot i, it must start at time t_i. Therefore, the following synchronization constraints are used. Note that constraint (1a) already insures that $S_{iv} = 0$ if $Z_{iv} = 0$.

$$S_{iv} \leq t_i \qquad i \in T, v \in W \tag{13a}$$

$$S_{iv} \geq t_i - H \cdot (1 - Z_{iv}) \qquad i \in T, v \in W \tag{13b}$$

4.5. MOS-FST Model

In the MOS-FST model, the time-points variables t_i of the MOS-SST model are used and are fixed a priori. They can therefore be considered as parameters. In this paper, these time-points are always selected using a uniform time discretization.

$$t_i = \frac{i-1}{n} \cdot H \qquad i \in T \tag{14}$$

Synchronization Constraint. If operation v is assigned to priority-slot i, it must start at time t_i. Therefore, the following synchronization constraints are used.

$$S_{iv} = t_i \cdot Z_{iv} \qquad i \in T, v \in W \tag{15}$$

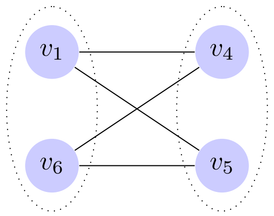


Figure 4: Biclique $(\{v_1, v_6\}; \{v_4, v_5\})$.

4.6. SOS Model

All constraints from the MOS model are still valid for the SOS model. However, a new assignment constraint is defined.

Assignment Constraint. At most one operation has to be assigned to each priority-slot.

$$\sum_{v \in W} Z_{iv} \leq 1 \qquad i \in T \qquad (16)$$

5. Strengthened Reformulations

The models presented in the previous section may not be effectively solved by MILP solvers. Indeed, special attention needs to be paid to their LP relaxation. As MILP solvers usually perform better when the model has a tight LP relaxation, we introduce strengthened formulations based on the non-overlapping structure of the problem.

5.1. Non-overlapping Graph Properties

In this section, we recall the definition of a *clique* and a *biblique* in the context of the non-overlapping graph G_{NO}. A *clique* of G_{NO} is a subset of the set of operations $W' \subset W$ such that any two operations in W' must not overlap. A *maximal clique* is a clique that is not a subset of any other clique. An *isolated clique* is a clique that has no edges connecting it to its complement in the graph. For the case-study, the non-overlapping graph displayed in Figure 2 contains 9 non-maximal cliques of two vertices, one for each non self-loop edge (e.g. $\{v_1, v_4\}$). It contains 4 maximal cliques of three vertices $\{v_1, v_4, v_5\}$, $\{v_2, v_4, v_6\}$, $\{v_3, v_5, v_6\}$, $\{v_4, v_5, v_6\}$. It contains no clique of larger size and no isolated clique.

We define a *biclique* of G_{NO} as a pair of sets of operations $(W_1; W_2) \in W^2$ such that for any pair of operations $(v_1; v_2) \in W_1 \times W_2$, $\{v_1, v_2\}$ is an edge of G_{NO}. Sets W_1 and W_2 are not necessarily disjoint. A *maximal biclique* of G_{NO} is a biclique that is not contained in any other biclique of G_{NO}. For the case-study, the non-overlapping graph displayed in Figure 2 contains 9 non-maximal bicliques of two vertices, one for each non self-loop edge (e.g. $(W_1 = \{v_1\}; W_2 = \{v_4\})$). It contains 4 maximal bicliques of three vertices that can be derived from its maximal cliques $(\{v_1, v_4, v_5\}; \{v_1, v_4, v_5\})$, $(\{v_2, v_4, v_6\}; \{v_2, v_4, v_6\})$, $(\{v_3, v_5, v_6\}; \{v_3, v_5, v_6\})$, and $(\{v_4, v_5, v_6\}; \{v_4, v_5, v_6\})$. It also contains 3 maximal cliques composed of four vertices $(\{v_1, v_6\}; \{v_4, v_5\})$, $(\{v_2, v_5\}; \{v_4, v_6\})$, $(\{v_3, v_4\}; \{v_5, v_6\})$ and 3 maximal bicliques composed of five vertices $(\{v_4\}; \{v_1, v_2, v_4, v_5, v_6\})$, $(\{v_5\}; \{v_1, v_3, v_4, v_5, v_6\})$, $(\{v_6\}; \{v_2, v_3, v_4, v_5, v_6\})$. Figure 4 depicts the subgraph of G_{NO} corresponding to biclique $(\{v_1, v_6\}; \{v_4, v_5\})$.

Remark 4: Isolated cliques are always maximal as they cannot be extended to larger cliques.

Remark 5: Using maximal cliques instead of non-maximal cliques generally leads to a tighter LP relaxation. Constraints based on cliques will therefore be applied to maximal cliques only. Appendix A shows how applying strengthened constraints to maximal cliques only generates the tightest and most compact model. The same remark applies to bicliques.

Remark 6: The number of maximal cliques in an undirected graph might be exponential in the size of the graph. Nevertheless, there are exponential-time algorithms to enumerate all maximal cliques of a graph and we assume the non-overlapping graph is small enough so that this task can be performed in reasonable time. The same remark applies to bicliques.

5.2. MOS Model

Aggregated Assignment Constraint. Let $W' \subset W$ be a clique of the non-overlapping graph G_{NO}. Then, at most one operation from W' can be assigned to priority-slot i. Therefore, the following constraint, which is at least as strong as constraint (4) as shown in Appendix A, is valid.

$$\sum_{v \in W'} Z_{iv} \leq 1 \qquad i \in T, W' \in \mathit{clique}(G_{NO}) \tag{17}$$

Aggregated Non-overlapping Constraint. Given a clique $W' \subset W$ of the non-overlapping graph G_{NO}, the following aggregated non-overlapping constraint, which is at least as strong as constraint (6) as shown in Appendix A, is valid. Note that only one term in each summation can be non-zero due to the aggregated assignment constraint (17).

$$\sum_{v \in W'} E_{i_1 v} \leq \sum_{v \in W'} S_{i_2 v} + H \cdot (1 - \sum_{v \in W'} Z_{i_2 v}) \qquad i_1, i_2 \in T, i_1 < i_2, W' \in \mathit{clique}(G_{NO}) \tag{18}$$

Aggregated Non-overlapping Constraint with Intermediate Operations. Given a clique $W' \subset W$ of the non-overlapping graph G_{NO}, and two priority-slots $i_1, i_2 \in T$ such that $i_1 < i_2$, constraint (18) can be further strengthened (see Appendix A) by including the duration of operations assigned to intermediate priority-slots i ($i_1 < i < i_2$).

$$\sum_{v \in W'} E_{i_1 v} + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W'} D_{iv} \leq \sum_{v \in W'} S_{i_2 v} + H \cdot (1 - \sum_{v \in W'} Z_{i_2 v}) \qquad i_1, i_2 \in T, i_1 < i_2, W' \in \mathit{clique}(G_{NO}) \tag{19}$$

Aggregated Non-overlapping Constraint with Clique Transition Times. Given a clique $W' \subset W$ of the non-overlapping graph G_{NO}, the clique transition time $TR_{W'}$ is the minimum time delay between any two executions of operations in W'. If W' represents a set of operations executed in a unit, $TR_{W'}$ corresponds to a unit transition time. Clique transition times can be enforced as follows. Note that the value of the big-M constant is increased to a new valid upper bound for the left hand side of the inequality: $H + TR_{W'}$.

$$\sum_{v \in W'} (E_{i_1 v} + TR_{W'} \cdot Z_{i_1 v}) + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W'} (D_{iv} + TR_{W'} \cdot Z_{iv}) \leq \sum_{v \in W'} S_{i_2 v} + (H + TR_{W'}) \cdot (1 - \sum_{v \in W'} Z_{i_2 v}) \tag{20}$$
$$i_1, i_2 \in T, i_1 < i_2, W' \in \mathit{clique}(G_{NO})$$

5.3. MOS-SST Model

Aggregated Synchronization Constraints. Given a clique $W' \subset W$ of the non-overlapping graph G_{NO}, the following aggregated synchronization constraints, which are tighter than constraints (13) as shown in Appendix A, are valid. Note that in each summation only one term can be non-zero.

$$\sum_{v \in W'} S_{iv} \leq t_i \qquad i \in T, W' \in \mathit{clique}(G_{NO}) \tag{21a}$$

$$\sum_{v \in W'} S_{iv} \geq t_i - H \cdot (1 - \sum_{v \in W'} Z_{iv}) \qquad i \in T, W' \in \mathit{clique}(G_{NO}) \tag{21b}$$

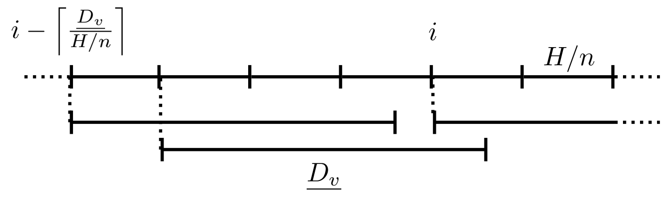


Figure 5: Assignment constraint using consecutive time-points.

Tightening Precedence Constraint. Consider a precedence constraint between sets of operations W_1 and W_2 ($P_{W_1W_2} = 1$). Assume that an operation from W_1 is assigned to priority-slot i_1 with associated time-point t_1 and that an operation from W_2 is assigned to priority-slot i_2 with associated time-point t_2. Then, due to the precedence constraint, we must have $t_1 < t_2$. Therefore, a necessary condition for the precedence constraint to hold true is $i_1 < i_2$. The following constraint insures that this condition is satisfied, and therefore complements constraint (10).

$$\sum_{\substack{j\in T\\ j<i}} \sum_{v\in W_1} Z_{jv} \geq \sum_{\substack{j\in T\\ j\leq i}} \sum_{v\in W_2} Z_{jv} \qquad i \in T, W_1, W_2 \subset W, P_{W_1W_2} = 1 \tag{22}$$

5.4. MOS-FST Model

Assignement Constraint using Time-points. Consider an operation $v \in W$. If the time interval between two time-points is smaller than the duration of v, then v cannot start at both time-points (v cannot be assigned to both priority-slots). As illustrated in Figure 5, v cannot be assigned to priority-slots $i - \left\lceil \frac{\underline{D_v}}{H/n} \right\rceil + 1$ and i simultaneously, as well as any other priority-slot in between. However, it can be assigned to priority-slots $i - \left\lceil \frac{\underline{D_v}}{H/n} \right\rceil$ and i simultaneously. Note that the lower bound on the duration of v is used to account for the case of variable processing times. Therefore, the following constraint, which is a stronger extension of constraint (4), is valid.

$$Z_{iv} + \sum_{\substack{j\in T\\ i-\delta_v<j<i}} Z_{jv} \leq 1 \qquad i \in T, v \in W, \delta_v = \left\lceil \frac{\underline{D_v}}{H/n} \right\rceil \tag{23}$$

Aggregated Assignement Constraint. The previous constraint (23) can be extended to cliques of G_{NO}.

$$\sum_{v\in W'} \left\{ Z_{iv} + \sum_{\substack{j\in T\\ i-\delta_{W'}(v)<j<i}} Z_{jv} \right\} \leq 1 \qquad i \in T, W' \in clique(W), \delta_v = \left\lceil \frac{\underline{D_v}}{H/n} \right\rceil \tag{24}$$

Aggregated Assignement Constraint with Transitions Times. The previous constraint (24) can also be extended to the case of clique transition times.

$$\sum_{v\in W'} \left\{ Z_{iv} + \sum_{\substack{j\in T\\ i-\delta_{W'}(v)<j<i}} Z_{jv} \right\} \leq 1 \qquad i \in T, W' \in clique(W), \delta_{W'}(v) = \left\lceil \frac{\underline{D_v} + TR_{W'}}{H/n} \right\rceil \tag{25}$$

5.5. SOS Model

Aggregated Non-overlapping Constraint. Additional aggregated non-overlapping constraints can be generated based on bi-cliques of the non-overlapping graph for the SOS model. Given a biclique (W_1, W_2) of the non-overlapping graph G_{NO}, the following aggregated non-overlapping constraints, which are at least as strong as constraints (5) as shown in Appendix A, are valid. Note that in each summation only one term can be non-zero due to the SOS specific assignment constraint (16). Also, note that constraints (18) and (26) are not redundant as they are applied to different sets of operations.

$$\sum_{v\in W_1} E_{i_1 v} \leq \sum_{v\in W_2} S_{i_2 v} + H \cdot (1 - \sum_{v\in W_2} Z_{i_2 v}) \qquad i_1, i_2 \in T, i_1 < i_2, (W_1, W_2) \in \mathit{maxbiclique}(G_{NO}) \tag{26a}$$

$$\sum_{v\in W_2} E_{i_1 v} \leq \sum_{v\in W_1} S_{i_2 v} + H \cdot (1 - \sum_{v\in W_1} Z_{i_2 v}) \qquad i_1, i_2 \in T, i_1 < i_2, (W_1, W_2) \in \mathit{maxbiclique}(G_{NO}) \tag{26b}$$

6. Solution Methods

Determining the minimum number of priority-slots needed to find the optimal schedule is non-trivial. A commonly used algorithm is to solve several scheduling models, each time increasing the number of priority-slots. We present two different approaches based on this idea.

6.1. Additive Approach

The MOS, MOS-SST and SOS time representations are such that any solution found with n priority-slots can be found with $n+1$ priority-slots: $z \in \mathbf{MOS}(n) \Rightarrow z \in \mathbf{MOS}(n+1)$. Therefore, the corresponding models can be solved by successively increasing by 1 the number of priority-slot. The additive approach is described by Algorithm 1. The parameter n_0 is the initial number of priority-slots. To improve efficiency of the branch & bound algorithm at each iteration, the cutoff parameter is set using the objective value of the best incumbent found so far. During some iterations, the MILP model may not have any solution strictly better than the best incumbent. In such cases, it will not return any solution even though the model may be feasible.

In this paper, three stopping criteria will be used. The first one is $(\Delta \leq \varepsilon)$ where ε is an absolute tolerance on the variation of objective value. In general, this criterion does not guarantee global optimality of the solution even when setting ε to 0. However, it leads to a small number of iterations. Also, in all our experiments, this stopping criterion always returned the global optimal solution, which was only proved optimal by using the following stopping criterion.

The second stopping criterion is $(n > \overline{n})$ where $\overline{n}$ is an upper limit on the number of priority-slots. In some cases, it is possible to determine an upper limit on the number of priority-slots needed to find the optimal solution of the scheduling problem. In such cases, this stopping criterion guarantees global optimality of the solution.

The third stopping criterion is a time limit on the total computational time, which of course does not guarantee global optimality.

Algorithm 1: Additive approach.

```
begin
    $z^* \longleftarrow \emptyset$ ;
    cutoff $\longleftarrow -\infty$ ;
    $n \longleftarrow n_0$ ;
    repeat
        $z \longleftarrow$ Maximize($\boldsymbol{MOS(n)}$, cutoff) ;
        $\Delta \longleftarrow z$.objval() $- z^*$.objval() ;
        if $\Delta > 0$ then
            $z^* \longleftarrow z$ ;
            cutoff $\longleftarrow z^*$.objval() ;
        $n \longleftarrow n + 1$ ;
    until stopping condition ;
    return $z^*$ ;
end
```

It is important to note that at each iteration, the integer feasible space of **MOS**(n) includes scheduling solutions explored during previous iterations. In order to avoid redundant search, we introduce constraint (27) that rejects any solution that do not make use of all priority-slots. By adding this constraint to the **MOS**(n) model, the property $z \in \mathbf{MOS}(n) \Rightarrow z \in \mathbf{MOS}(n+1)$ is no longer valid. The same remark holds true for the **MOS-SST**(n) and **SOS**(n) models.

$$\sum_{v \in W} Z_{iv} \geq 1 \qquad\qquad i \in T \tag{27}$$

6.2. Multiplicative Approach

The previous approach could be used to solve a scheduling problem using the MOS-FST time representation. However, a major flaw is that it is not guaranteed that a solution found with n priority-slots can be found with $n+1$ priority-slots. This can be overcome by multiplying the number of priority-slots be a factor of 2 instead of using an increment. Indeed, any solution found with n priority-slots can be found with $2n$ priority-slots: $z \in \mathbf{MOS\text{-}FST}(n) \Rightarrow z \in \mathbf{MOS\text{-}FST}(2n)$. The multiplicative approach is therefore very similar to the additive approach and is described by Algorithm 2. The stopping criteria introduced for the additive approach can still be used for the multiplicative approach.

Algorithm 2: Multiplicative approach.

```
begin
    $z^* \longleftarrow \emptyset$ ;
    cutoff $\longleftarrow -\infty$ ;
    $n \longleftarrow n_0$ ;
    repeat
        $z \longleftarrow$ Maximize($\boldsymbol{MOS(n)}$, cutoff) ;
        $\Delta \longleftarrow z$.objval() $- z^*$.objval() ;
        if $\Delta > 0$ then
            $z^* \longleftarrow z$ ;
            cutoff $\longleftarrow z^*$.objval() ;
        $n \longleftarrow 2 \cdot n$ ;
    until stopping condition ;
    return $z^*$ ;
end
```

6.3. Direct Approach

Although the additive approach can be used to solve SOS models, it might not be very efficient. Indeed, each time the number of priority-slot is increased by 1, the solver can only schedule one additional operation. In MOS or MOS-SST models, several additional operations can be scheduled at each iteration, which leaves much more flexibility to better improve the objective value. Instead, a direct approach can be used. It consists of choosing a fixed value for n and solving the MILP model once. In some cases, the total number of executions of operations is fixed and known in advanced, so it can be used as the number of priority-slots, guaranteeing global optimality of the solution obtained. It other cases, a detailed analysis of the scheduling structure of the problem needs to be performed to efficiently define a value for n (see Section 9). Global optimality may not be guaranteed if n is too small, or if additional constraints are used to improve the search.

7. Single-Stage Batch Scheduling Problem

In this section, we apply the various time representations to model single-stage batch scheduling problems. We study several instances introduced in Pinto and Grossmann (1995). The largest instance has 29 orders to be processed before given due dates from time 0 to time 30. Four units are available to process each single-stage order. Each unit can process only one order at a time and a minimum unit-specific set-up time is required between any two orders. Each order can only be processed on a subset of all units with order-and-unit-specific processing times. Table 2 displays all required data. For each order, units that do not have a corresponding processing time cannot be selected for this order. The objective is to minimize total earliness which corresponds to maximizing the end times of all orders. Five instances have been studied with 8, 12, 18, 25, and 29 orders, which we denote SSBSP8, ..., SSBSP29. We introduce the following specific sets.

- $O \subset \{o_1, \ldots, o_{29}\}$ is the set of orders ($O = \{o_1, \ldots, o_8\}$ for SSBSP8).
- $U = \{u_1, \ldots, u_4\}$ is the set of units.
- U_o is the set of units on which order $o \in O$ can be processed. For instance, $U_{o_1} = \{u_1, u_4\}$.
- O_u is the set of orders that can be processed on unit $u \in U$. For instance, $O_{u_3} = \{o_4, o_5, o_7, o_8\}$.

The set of operations W can then be defined as follows. Exactly one operation is defined for each order and each unit on which the order can be processed. This reduces the number of indices from 2 to 1, although the combinatorial size of the problem remains identical.

$$W = \{(o, u) \in O \times U, u \in U_o\} = \{o_1u_1, o_1u_4, o_2u_1, o_2u_4, o_3u_1, o_3u_4, o_4u_3, o_4u_4, \ldots\}$$

Figure 6 depicts the non-overlapping graph of SSBSP8. It contains 3 isolated cliques each corresponding to one unit (unit u_2 cannot be used in this instance as it is excluded for all orders, as seen in Table 2). We will denote $W_u = \{v = (o, u), o \in O_u\}$, where $u \in U$, the set of operations executed on unit $u \in U$, W_u is an isolated clique of G_{NO}. Given an order $o \in O$, exactly one execution of this order must be performed. Therefore, the set $W_o = \{v = (o, u), u \in U_o\}$ has an associated cardinality constraint of 1, that is $\underline{N_{W_o}} = \overline{N_{W_o}} = 1$.

7.1. MOS Model

The MOS model for the single-stage batch scheduling problem is derived from constraints (1)-(3), (17), (20), and (27).

Table 2: Single-Stage Batch Scheduling Example Data

Order	Due date (hr)	Unit processing time (hr)			
		1	2	3	4
Set-up times		0.180	0.175	0.000	0.237
1	15	1.538			1.194
2	30	1.500			0.789
3	22	1.607			0.818
4	25			1.564	2.143
5	20			0.736	1.017
6	30	5.263			3.200
7	21	4.865		3.025	3.214
8	26			1.500	1.440
9	30			1.869	2.459
10	29		1.282		
11	30		3.750		3.000
12	21		6.796	7.000	5.600
13	30	11.250			6.716
14	25	2.632			1.527
15	24	5.000			2.985
16	30	1.250			0.783
17	30	4.474			3.036
18	30		1.429		
19	13		3.130		2.687
20	19	2.424		1.074	1.600
21	30	7.317		3.614	
22	20			0.864	
23	12			3.624	
24	30			2.667	4.000
25	17	5.952		3.448	4.902
26	20	3.824			1.757
27	11	6.410			3.937
28	30	5.500			3.235
29	25				4.286

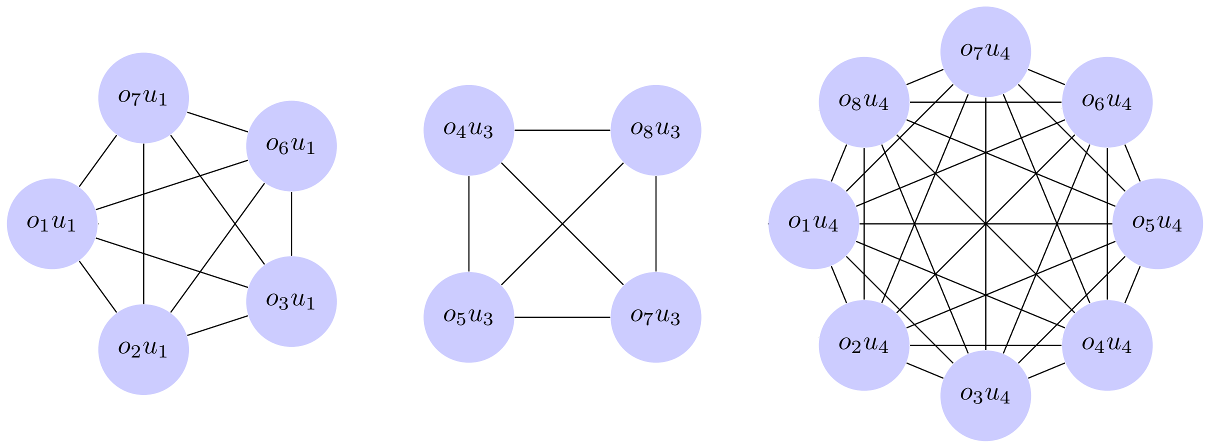


Figure 6: Non-overlapping graph with isolated cliques for SSBSP8.

$$\textbf{Maximize} \qquad \sum_{i \in T} \sum_{v \in W} E_{iv}$$

Subject to

$$D_{iv} = D_v \cdot Z_{iv} \qquad i \in T, v \in W$$

$$E_{iv} \leq \overline{E_v} \cdot Z_{iv} \qquad i \in T, v \in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i \in T, v \in W$$

$$\sum_{i \in T} \sum_{v \in W_o} Z_{iv} = 1 \qquad o \in O$$

$$\sum_{v \in W_u} Z_{iv} \leq 1 \qquad i \in T, u \in U$$

$$\sum_{v \in W_u} (E_{i_1 v} + TR_{W_u} \cdot Z_{i_1 v}) + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W_u} (D_{iv} + TR_{W_u} \cdot Z_{iv})$$
$$\leq \sum_{v \in W_u} S_{i_2 v} + (H + TR_{W_u}) \cdot (1 - \sum_{v \in W_u} Z_{i_2 v}) \qquad i_1, i_2 \in T, i_1 < i_2, u \in U$$

$$\sum_{v \in W} Z_{iv} \geq 1 \qquad i \in T$$

$$S_{iv}, D_{iv}, E_{iv} \geq 0 \qquad i \in T, v \in W$$

$$Z_{iv} \in \{0, 1\} \qquad i \in T, v \in W$$

This model can be strengthened in different ways. For each unit, it is possible to derive minimum and maximum of orders that will be processed on it. The maximum number of orders that can be process on unit $u \in U$ is defined as the number of operations in the set.

$$\overline{N_{W_u}} = |W_u| \qquad u \in U$$

An improved upper cardinality value for W_u can be obtained by considering the processing times of operations that might be processed on unit u. Indeed, due to a limited scheduling horizon, it might not be possible to execute all operations in W_u. Therefore, we use the following improved definition that solves a one-dimensional knapsack problem where all operation have a value of 1. A linear algorithm to solve this problem consists in ordering the operations in W_u with respect to their duration D_v and, starting from the operation with the lowest processing time, setting γ_v to 1 until the knapsack limit $H + TR_{W_u}$ is reached. For the remaining operations, γ_v is set to 0.

$$\overline{N_{W_u}} = \max \left\{ \sum_{v \in W_u} \gamma_v \,\middle|\, \gamma_v \in \{0,1\}, \sum_{v \in W_u} (D_v + TR_{W_u}) \cdot \gamma_v \leq H + TR_{W_u} \right\} \qquad u \in U$$

We denote $W_u^1 = \{v = (o, u) \in W, U_o = \{u\}\}$ the set of operations that can only be processed on unit $u \in U$. The minimum number of orders that will be processed on unit $u \in U$ can be defined as follows.

$$\underline{N_{W_u}} = |W_u^1| \qquad u \in U$$

However, as the number of orders executed on units different than $u \in U$ is limited, $\underline{N_{W_u}}$ can be increased as follows.

$$\underline{N_{W_u}} = \max \left\{ |W_u^1|, |O| - \sum_{\substack{u' \in U \\ u' \neq u}} \min\{n, \overline{N_{W_{u'}}}\} \right\} \qquad u \in U$$

Table 3: Unit cardinality bounds depending on parameter n for SSBSP29

n	u_1	u_2	u_3	u_4
8	[8,8]	[5,5]	[8,8]	[8,8]
9	[6,9]	[2,5]	[6,9]	[6,9]
10	[4,9]	[2,5]	[5,10]	[5,10]
11	[2,9]	[2,5]	[4,11]	[4,11]
12	[1,9]	[2,5]	[3,11]	[4,12]
13	[0,9]	[2,5]	[2,11]	[4,13]
14	[0,9]	[2,5]	[2,11]	[4,14]

The values obtained for $\underline{N_{W_u}}$ and $\overline{N_{W_u}}$ in problem SSBSP29 are displayed in Table 3 as an example. Using $\underline{N_{W_u}}$ and $\overline{N_{W_u}}$, the following cardinality constraint can be derived.

$$\underline{N_{W_u}} \leq \sum_{i \in T} \sum_{v \in W_u} Z_{iv} \leq \overline{N_{W_u}} \qquad u \in U \tag{28}$$

Given a unit $u \in U$, W_u is an isolated clique of G_{NO}, so the optimal sequence of operations from W_u is not affected by operations from $W \setminus W_u$. It is therefore possible to reduce the set of possible sequences of operations from W_u by assigning these operations to the first priority-slots only. In other words, an operation W_u can be assigned to priority-slot i only if an operation from W_u is assigned to priority-slot $i-1$. This leads to the following symmetry-breaking constraint.

$$\sum_{v \in W_u} Z_{iv} \leq \sum_{v \in W_u} Z_{(i-1)v} \qquad i \in T, i \neq 1, u \in U \tag{29}$$

This idea can be further applied to other constraints in the model. A maximum cardinality constraint on the set of operations W_u can be enforced by setting to 0 assignment variables corresponding to the last priority-slots for this set.

$$\sum_{v \in W_u} Z_{iv} = 0 \qquad i \in T, i > \overline{N_{W_u}}, u \in U \tag{30}$$

Also, a minimum cardinality constraint on W_u can be enforced by assigning exactly one operation the first priority-slots.

$$\sum_{v \in W_u} Z_{iv} = 1 \qquad i \in T, i \leq \underline{N_{W_u}}, u \in U \tag{31}$$

Besides, non-overlapping constraint (20) can be tightened for the first priority-slots as follows.

$$\sum_{v \in W_u} (E_{i_1 v} + TR_{W_u} \cdot Z_{i_1 v}) + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W_u} (D_{iv} + TR_{W_u} \cdot Z_{iv}) \leq \sum_{v \in W_u} S_{i_2 v} \tag{32}$$
$$i_1, i_2 \in T, i_1 < i_2 \leq \underline{N_{W_u}}, u \in U$$

The single-stage batch scheduling problems are solved with the MOS model plus constraints (28)-(32). The additive approach is used with the second stopping criterion ($n > \max_u \overline{N_u}$) which guarantees global optimality of the solution. Indeed, for each unit $u \in U$, no more than $\overline{N_u}$ priority-slots are needed to sequence operations on it. The initial number of priority-slots is set to $n_0 = \lceil |O| / |\{u | W_u \neq \emptyset\}| \rceil$ which is the minimum number of orders at least one unit has to process. Computational results are given in Table 4. Experiments were run on an Intel Xeon 1.86GHz processor using GAMS/CPLEX 11. For each iteration, the LP relaxation, MILP solution, number of nodes, CPU time and cumulative CPU time are displayed. The term "no solution" for MILP solution is used when CPLEX did not find any solution with higher objective value than the cutoff value which does not mean that no solution exists. For each problem, the global optimal solution is underlined. The CPU time elapsed until the first stopping criterion ($\Delta \leq 0$) is satisfied is also underlined. The results show that the first stopping criterion leads to significant reduction of CPU times compared to the second. Also, it is interesting to note the problem with

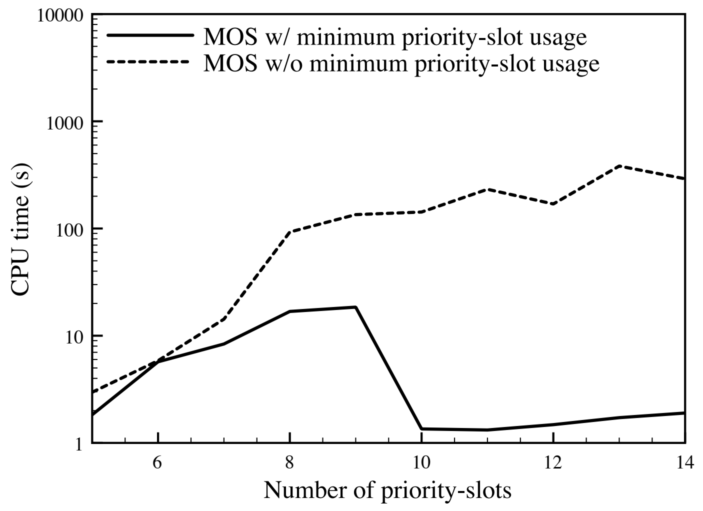


Figure 7: Effect of the minimum priority-slot usage constraint.

29 orders, although larger, is solved faster than the 25 orders problem. As the scheduling horizon is fixed to 30 hours, the density of operations is higher with 29 orders. Therefore, this problem is more constrained and has potentially fewer solution. The model size is increased but the branch & bound tree is smaller, thus leading to smaller CPU times. This is verified by the fact that fewer nodes are explored.

Additionally, we solved problem SSBSP18 using the full MOS models with or without the minimum priority-slot usage constraint (27). The results obtained with the additive approach are displayed in Figure 7. It shows that constraint (27) greatly reduces the search space at each iteration. In particular, using this constraint, the last iterations are solved in few seconds, whereas CPU times are higher than 100 seconds if it is not used.

We also solved the SSBSP18 instance with exactly 5 priority-slots using the full MOS model without and with symmetry-breaking constraints, which corresponds to the first iteration of the additive algorithm for this problem. Without symmetry-breaking constraints, the problem was solved in 115 seconds (124788 nodes) as opposed to less than 2 seconds (513 nodes) if symmetry-breaking constraints (29)-(32) are used. This shows why using symmetry-breaking concepts is crucial to solve scheduling problems as they tend to be highly degenerate (Kallrath, 2002).

We performed another experiment which consisted in solving the full MOS model by using the original assignment constraint (4) instead of the strengthened assignment constraint (17). Instance SSBSP18 was solved in 3.43 seconds (587 nodes) instead of 1.83 seconds (513 nodes) for the strengthened constraint. Thus, this constraint slightly helps to improve the solution time although CPLEX is able to generate cuts to tighten the LP relaxation accordingly. However, we also solved the full MOS model replacing the strengthened non-overlapping constraints (20) and (32) by the following non-overlapping constraint with unit-dependent transition time, which is based on constraint (8).

$$\begin{aligned} E_{i_1 v_1} + E_{i_1 v_2} + TR_{W_u} \cdot (Z_{i_1 v_1} + Z_{i_1 v_2}) \leq S_{i_2 v_1} + S_{i_2 v_2} + (H + TR_{W_u}) \cdot (1 - Z_{i_2 v_1} - Z_{i_2 v_2}) \\ i_1, i_2 \in T, i_1 < i_2, u \in U, v_1, v_2 \in W_u \end{aligned} \tag{33}$$

The model was solved in 3288 seconds and 1,078,034 nodes (vs 1.83 seconds and 513 nodes) which proves that CPLEX was not successful at improving the LP relaxation of the model for this mixed-integer constraint. In general, MILP solvers are very efficient at solving IPs as they are able to exploit the structure of the model in order generate very tight cuts (such as clique constraints, Nemhauser and Wolsey (1999)) that can lead to large improvements of the branch & bound search. However, it is much harder to generate tightening cuts from constraint (33) using the clique structure. Therefore, using strengthened formulations for mixed-integer constraints is very important as it greatly improves the performance of MILP

Table 4: MOS computational results for Single-Stage Batch Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
SSBSP8	3	189.000	189.000	3	0.81s	0.81s
	4	189.000	no solution	0	0.77s	1.58
	5	189.000	no solution	0	0.80s	2.38s
	6	189.000	no solution	0	0.76s	3.14s
	7	189.000	no solution	0	0.86s	4.00s
	8	188.035	no solution	0	0.90s	4.90s
SSBSP12	3	298.517	296.543	0	0.83s	0.83s
	4	299.000	297.974	604	1.67s	2.50s
	5	299.000	no solution	207	1.38s	3.88s
	6	299.000	no solution	198	1.71s	5.59s
	7	299.000	no solution	0	1.17s	6.76s
	8	299.000	no solution	0	1.02s	7.78s
	9	no solution			0.95s	8.73s
	10	no solution			1.01s	9.74s
	11	no solution			1.09s	10.83s
SSBSP18	5	461.563	450.760	513	1.83s	1.83s
	6	463.993	450.982	1952	5.71s	7.54s
	7	467.196	451.504	2924	8.37s	15.91s
	8	467.966	no solution	4600	16.84s	32.75s
	9	467.966	no solution	4600	18.47s	51.22
	10	459.877	no solution	0	1.35s	52.57
	11	no solution			1.32s	53.89s
	12	no solution			1.48s	55.37s
	13	no solution			1.72s	57.09s
	14	no solution			1.90s	58.99s
SSBSP25	7	593.615	575.929	3444	17.24s	17.24s
	8	601.884	579.089	18261	69.90s	87.14s
	9	607.713	579.570	55046	189.66s	276.80s
	10	609.000	no solution	77800	408.97s	685.77s
	11	609.000	no solution	22400	113.60s	799.37s
	12	597.717	no solution	0	3.61s	802.98s
	13	583.289	no solution	0	2.31s	805.29s
	14	no solution			2.54s	807.83s
SSBSP29	8	649.531	628.413	1583	12.97s	12.97s
	9	656.403	634.964	10870	56.16s	69.13s
	10	662.720	635.104	42249	206.66s	275.79s
	11	666.512	no solution	22400	110.83s	386.62s
	12	666.552	no solution	3700	32.72s	419.34s
	13	668.164	no solution	1800	28.33s	447.67s
	14	653.051	no solution	0	4.72s	452.39s

solvers.

7.2. MOS-SST Model

The MOS-SST model for the single-stage batch scheduling problem is derived from constraints (1)-(3), (12), (17), (20), (21), and (27).

$$\textbf{Maximize} \quad \sum_{i\in T}\sum_{v\in W} E_{iv}$$

Subject to

$$D_{iv} = D_v \cdot Z_{iv} \qquad i\in T, v\in W$$

$$E_{iv} \le \overline{E_v} \cdot Z_{iv} \qquad i\in T, v\in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i\in T, v\in W$$

$$\sum_{i\in T}\sum_{v\in W_o} Z_{iv} = 1 \qquad o\in O$$

$$\underline{N_u} \le \sum_{i\in T}\sum_{v\in W_u} Z_{iv} \le \overline{N_u} \qquad u\in U$$

$$\sum_{v\in W_u} Z_{iv} \le 1 \qquad i\in T, u\in U$$

$$\sum_{v\in W_u}(E_{i_1v} + TR_{W_u}\cdot Z_{i_1v}) + \sum_{\substack{i\in T\\ i_1<i<i_2}}\sum_{v\in W_u}(D_{iv} + TR_{W_u}\cdot Z_{iv}) \le \sum_{v\in W_u} S_{i_2v} + (H + TR_{W_u})\cdot(1 - \sum_{v\in W_u} Z_{i_2v}) \qquad i_1, i_2\in T, i_1<i_2, u\in U$$

$$t_{i-1} \le t_i \qquad i\in T, i\ne 1$$

$$\sum_{v\in W_u} S_{iv} \le t_i \qquad i\in T, u\in U$$

$$\sum_{v\in W_u} S_{iv} \ge t_i - H\cdot(1 - \sum_{v\in W'} Z_{iv}) \qquad i\in T, u\in U$$

$$\sum_{v\in W} Z_{iv} \ge 1 \qquad i\in T$$

$$S_{iv}, D_{iv}, E_{iv} \ge 0 \qquad i\in T, v\in W$$

$$Z_{iv} \in \{0,1\} \qquad i\in T, v\in W$$

$$t_i \in [0, H] \qquad i\in T$$

Due to the use of time-points, it is not possible to develop symmetry-breaking constraints based on the isolated cliques of G_{NO} for this model. Similarly to the MOS model, the additive approach is used and the initial number of priority-slots is set to $n_0 = \lceil |O| / |\{u|W_u \ne \emptyset\}| \rceil$. To solve the problem to global optimality, we use the second criterion ($n > |O|$). However, all instances except the first one are very expensive to solve, so we used a time limit of 1000 seconds. Computational results are given in Table 5. For each problem, the MOS-SST solution obtained with n priority-slots has lower objective value than the MOS solution obtained with the same number of priority-slots as stated previously. Overall results show that the MOS-SST time representation is much less effective than the MOS time representation.

Table 5: MOS-SST computational results for Single-Stage Batch Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
SSBSP8	3	189.000	177.001	1220	1.47s	1.47s
	4	189.000	183.149	6079	3.48s	4.95s
	5	189.000	185.567	16339	7.44s	12.39s
	6	189.000	187.815	8127	10.96s	23.35s
	7	189.000	188.823	25640	18.38s	41.73s
	8	189.000	$\underline{189.000}$	649	2.64s	$\underline{44.37s}$
SSBSP12	3	297.770	272.902	33	0.86s	0.86s
	4	299.000	288.031	24452	19.76s	20.62s
	5	299.000	290.640	195160	145.87s	166.49s
	6	299.000	292.697	507940	394.78s	561.27s
	7	299.000	294.289	815078	716.90s	1278.17s
SSBSP18	5	468.000	428.544	256375	476.74s	476.74s
	6	468.000	433.583	360510	+1000s	+1476.74s
SSBSP25	7	609.000	538.538	103912	+1000s	+1000s
SSBSP29	8	638.975	569.580	140638	+1000s	+1000s

7.3. MOS-FST Model

The MOS-FST model for the single-stage batch scheduling problem is derived from constraints (1)-(3), (15), and (25). Note that non-overlapping constraint (20) is not included in the model as assignment constraint (25) is sufficient to enforce non-overlapping constraints when processing times are constant ($\underline{D_v} = \overline{D_v}$). Also, constraint (27) is not included in the model as valid schedules containing periods with no activity would become infeasible.

$$\textbf{Maximize} \qquad \sum_{i\in T}\sum_{v\in W} E_{iv}$$

Subject to

$$D_{iv} = D_v \cdot Z_{iv} \qquad i \in T, v \in W$$

$$E_{iv} \le \overline{E_v} \cdot Z_{iv} \qquad i \in T, v \in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i \in T, v \in W$$

$$\sum_{i\in T}\sum_{v\in W_o} Z_{iv} = 1 \qquad o \in O$$

$$\underline{N_u} \le \sum_{i\in T}\sum_{v\in W_u} Z_{iv} \le \overline{N_u} \qquad u \in U$$

$$S_{iv} = t_i \cdot Z_{iv} \qquad i \in T, v \in W$$

$$\sum_{v\in W_u}\left\{ Z_{iv} + \sum_{\substack{j\in T\\ i-\delta_u(v)<j<i}} Z_{jv} \right\} \le 1 \qquad i \in T, u \in U, \delta_u(v) = \left\lceil \frac{D_v + TR_u}{H/n} \right\rceil$$

$$S_{iv}, D_{iv}, E_{iv} \ge 0 \qquad i \in T, v \in W$$

$$Z_{iv} \in \{0,1\} \qquad i \in T, v \in W$$

$$t_i = \frac{i-1}{n} \cdot H \qquad i \in T$$

Similarly to the MOS-SST model, it is not possible to develop symmetry-breaking constraints based on the isolated cliques of G_{NO} for this model. To solve it, the multiplicative approach is used and the initial number of priority-slots is set to $n_0 = \frac{H}{2} = 15$. Due to memory limitations we used the second stopping criterion ($n > 1920$) which does not guarantee

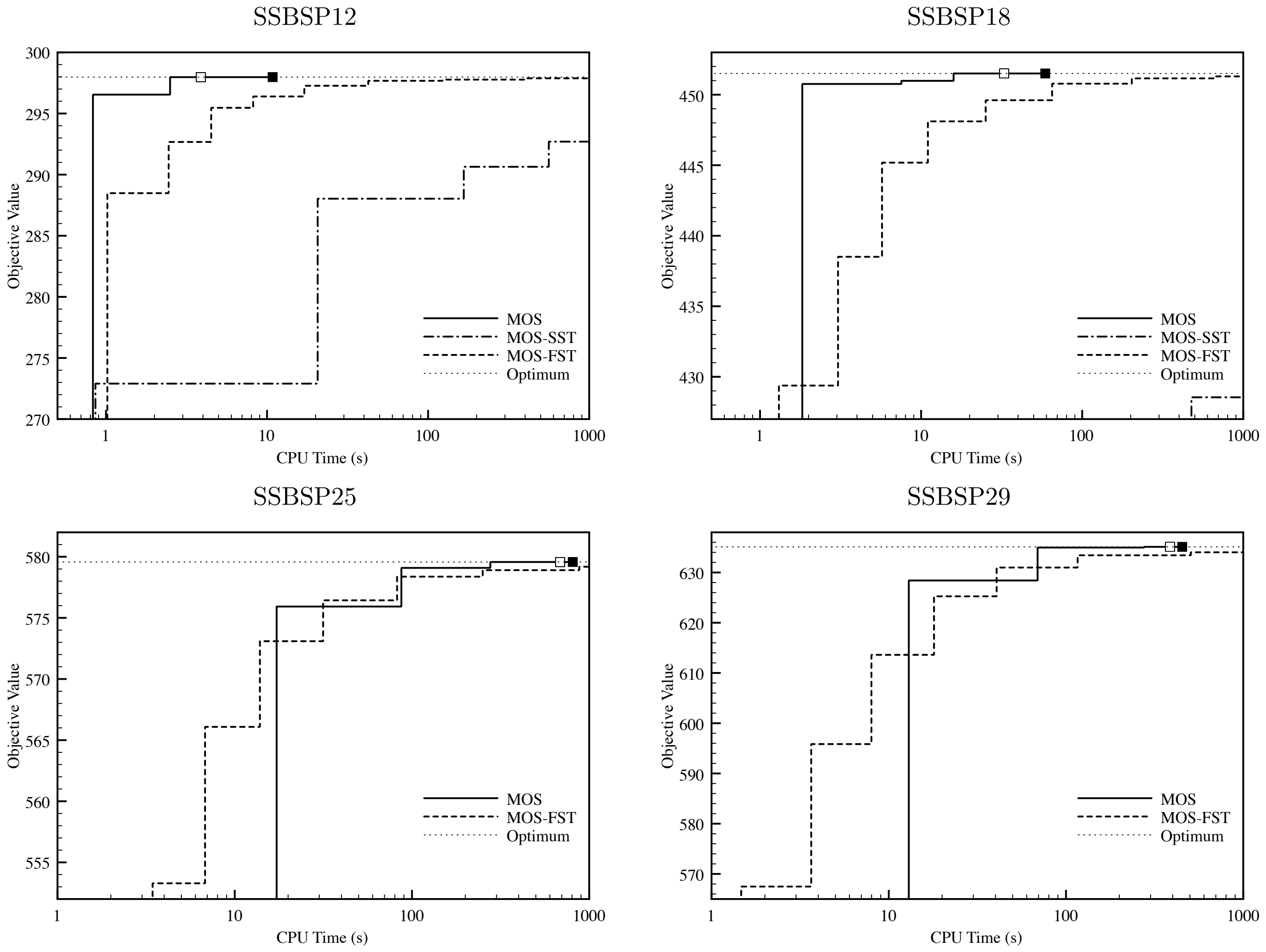


Figure 8: Comparison of the time representation for Single-Stage Batch Scheduling Problems.

global optimality of the solution. Computational results are given in Table 6. The first iterations are always computationally inexpensive and lead to good feasible solutions, which are improved in the next iterations. Although near-optimal solutions are obtained quickly, the gap is never entirely closed due to the time discretization. It is also interesting to note that most problems are solved at the root node.

7.4. Models Comparison

Figure 8 shows how the objective value of the best incumbent varies over time. A step increase of the objective value corresponds to a new solution found during the solution algorithm. Note that we do not include intermediate solutions found during each branch & bound search. Empty squares represent the time when the first stopping criterion is satisfied while plain squares represent the time when the second stopping criterion is satisfied. This applies only to the MOS model. From this figure, it is clear that the MOS model is superior to the other models as it finds first feasible solutions very fast, and is able to find optimal solutions and prove their optimality in reasonable time. The MOS-FST model compares well in particular for finding first feasible solutions. It also finds near-optimal solutions in reasonable time although it was slower than the MOS model for the instances that were considered. Also, it is clear that the MOS-SST representation is not well suited to solve these problems.

Table 6: MOS-FST computational results for Single-Stage Batch Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
SSBSP8	15	182.581	182.581	0	0.80s	0.80s
	30	186.013	186.013	0	1.07s	1.87s
	60	187.456	187.456	0	1.56s	3.43s
	120	188.594	188.594	0	2.77s	6.20s
	240	188.719	188.719	0	6.07s	12.27s
	480	188.813	188.813	0	16.89s	29.16s
	960	188.907	188.907	0	52.92s	82.08s
	1920	188.954	188.954	0	183.32s	265.40s
SSBSP12	15	288.482	288.482	0	1.02s	1.02s
	30	292.671	292.671	0	1.43s	2.45s
	60	295.466	295.466	0	2.05s	4.50s
	120	296.393	296.393	0	3.70s	8.20s
	240	297.268	297.268	0	8.83s	17.03s
	480	297.675	297.675	0	25.44s	42.47s
	960	297.772	297.772	0	80.46s	122.93s
	1920	297.878	297.878	0	282.79s	405.72s
SSBSP18	15	429.377	429.377	0	1.31s	1.31s
	30	438.507	438.507	0	1.74s	3.05s
	60	445.652	445.182	0	2.67s	5.72s
	120	448.808	448.107	0	5.30s	11.02s
	240	450.308	449.607	0	14.15s	25.17s
	480	451.339	450.782	0	39.90s	65.07s
	960	451.652	451.157	0	138.20s	203.27s
	1920	451.800	451.297	0	459.25s	662.52s
SSBSP25	15	537.852	536.490	0	1.39s	1.39s
	30	554.266	536.490	0	2.05s	3.44s
	60	566.344	566.085	0	3.36s	6.80s
	120	573.141	573.091	0	7.07s	13.87s
	240	576.435	576.435	0	17.66s	31.53s
	480	578.464	578.373	0	50.89s	82.42s
	960	578.996	578.904	0	168.31s	250.73s
	1920	579.300	579.185	0	628.46s	879.19s
SSBSP29	15	570.475	567.485	0	1.47s	1.47s
	30	597.973	595.846	0	2.18s	3.65s
	60	616.779	613.610	8	4.33s	7.98s
	120	627.674	625.258	5	10.02s	18.00s
	240	631.863	630.988	0	22.60s	40.60s
	480	634.336	633.426	0	75.74s	116.34s
	960	635.119	634.019	512	389.57s	505.91s
	1920	635.741	no solution	400	+1000s	+1505.91s

Table 7: Multi-Stage Batch Scheduling Example Data

Stage	Unit	Transition Time (hr)	Order processing time (hr)									
			1	2	3	4	5	6	7	8	9	10
1	1	8	18.1	23	18.1	20	17	15	31	12	13	12
	2	8	18.1	23	18.1	20	17	14	30	12	7	4
	3	8	18.1	23	18.1	20	17	13	34	14	8	23
	4	8	18.1	23	18.1	20	17	12	32	15	9	12
	5	8	18.1	23	18.1	20	17	18	31			
	6	8	18.1	23	18.1	20	17	15		16	16	14
2	7	8	14	14	14	11	14	15	31		15	13
	8	1	5	5	5	5	5	7	31	16	15	13
	9	1	5	5	5	5	5	7	31	15	11	13
3	10	2.5	12	12	24	12	12	13	14	12	13	12
	11	2.5	12	12	24	12	12	12	15	13	7	4
	12	2.5	12	12	24	12	12	15	16	14	8	23
	13	2.5	12	12	24	12	12	17	41	14	9	12
	14	2.5	12	12	24	12	12	17	15	14		
	15	2.5	12	12	24	12	12	18	81	14	16	14
	16	2.5	12	12	24	12	12	19		14	15	13
	17	2.5	12	12	24	12	12		16	14	15	13
	18	2.5		12				16	16	14	11	13
	19	2.5		12				13	21	10	12	6
4	20	6	9.5		9.3	7.9	12.5	13.5	12	10	15	
	21	6	9.5		9.3	7.9	12.5	14	13	9	17	12
	22	24		100				14.5	11	8	17	23
5	23	4	24		24	24	24	12	11		22	12
	24	4	24		24	24	24	12	11	7	21	22
	25	5		48				23	11	7		12

8. Multi-Stage Batch Scheduling Problem

We now extend the previous study to multi-stage batch scheduling problems from Pinto and Grossmann (1995). The largest instance has 8 orders to be processed in 5 consecutive stages before identical due dates $t = 500$ hr. Twenty-five units are available to process each order. Each unit is associated to one stage and can process only one order at a time. A minimum unit-specific set-up time is required between any two orders processed on the unit. Each order can only be processed on a subset of all units with order-and-unit-specific processing times. Table 7 displays all required data. For each order, units that do not have a corresponding processing time cannot be selected for this order. The objective, as introduced by Pinto and Grossmann (1995), is to minimize total weighted earliness which corresponds to maximizing a weighted summation of the end time of all orders in all stages. The weights are stage dependent and defined as $w_l = 0.2 \cdot l$. Three instances have been studied with 5, 8 and 10 orders, which we denote MSBSP5, MSBSP8 and MSBSP10. We introduce the following specific sets.

- $O \subset \{o_1, \ldots, o_8\}$ is the set of orders ($O = \{o_1, \ldots, o_5\}$ for MSBSP5).
- $L = \{l_1, \ldots, l_5\}$ is the set of stages
- $U = \{u_1, \ldots, u_{25}\}$ is the set of units.
- U_o is the set of units on which order $o \in O$ can be processed. For instance, $U_{o_1} = \{u_1, \ldots, u_{17}, u_{20}, u_{21}, u_{23}, u_{24}\}$.
- O_u is the set of orders that can be processed on unit $u \in U$. For instance, $O_{u_{20}} = \{o_1, o_3, \ldots, o_8\}$.
- l_u is the stage in which unit $u \in U$ can be used. For instance, $l_{u_7} = l_2$

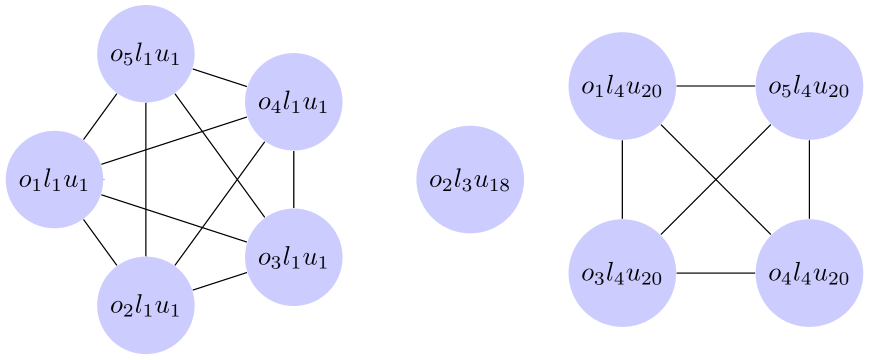


Figure 9: Partial non-overlapping graph with isolated cliques for MSBSP5.

- U_l is the set of units which can be used in stage $l \in L$. For instance, $U_{l_5} = \{u_{23}, u_{24}, u_{25}\}$

The set of operations can then be defined as follows. Exactly one operation is defined for each order, each stage and each unit on which the order can be processed. This reduces the number of indices from 3 to 1, although the combinatorial size of the problem remains identical.

$$W = \{(o, l, u) \in O \times L \times U, u \in U_o \cap U_l\} = \{o_1l_1u_1, \dots, o_1l_1u_6, o_1l_2u_7, \dots, o_1l_2u_9, o_1l_3u_{10}, \dots\}$$

Figure 9 partially depicts the non-overlapping graph of MSBSP5. It contains 25 isolated cliques each corresponding to one unit. We will denote $W_u = \{v = (o, l, u), o \in O_u, l = l_u\}$, where $u \in U$, the set of operations executed on unit u, W_u is an isolated clique of G_{NO}. Given an order $o \in O$ and a stage $l \in L$, exactly one execution of order o in stage l must be performed. Therefore, the set $W_{ol} = \{v = (o, l, u), u \in U_o \cap U_l\}$ has an associated cardinality constraint of 1, that is $\underline{N_{W_{ol}}} = \overline{N_{W_{ol}}} = 1$. Furthermore, due to the multi-stage feature of this problem, precedence constraints exist between operations belonging to different stages of the same order. More precisely, for each order $o \in O$ and each stage $l \in L \setminus \{l_1\}$, sets $W_{o(l-1)}$ and W_{ol} must satisfy a precedence constraint $P_{W_{o(l-1)}W_{ol}} = 1$.

8.1. MOS Model

The MOS model for the single-stage batch scheduling problem is derived from constraints (1)-(3), (10), (17), (20), (27), and (29)-(32).

Maximize $$\sum_{i\in T}\sum_{\substack{v\in W\\ v=(o,l,u)}} 0.2\cdot l\cdot E_{iv}$$

Subject to

$$D_{iv} = D_v\cdot Z_{iv} \qquad i\in T, v\in W$$

$$E_{iv} \le \overline{E_v}\cdot Z_{iv} \qquad i\in T, v\in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i\in T, v\in W$$

$$\sum_{i\in T}\sum_{v\in W_{ol}} Z_{iv} = 1 \qquad o\in O, l\in L$$

$$\underline{N_u} \le \sum_{i\in T}\sum_{v\in W_u} Z_{iv} \le \overline{N_u} \qquad u\in U$$

$$\sum_{i\in T}\sum_{v\in W_{o(l-1)}} E_{iv} \le \sum_{i\in T}\sum_{v\in W_{ol}} S_{iv} \qquad o\in O, l\in L, l\neq 1$$

$$\sum_{v\in W_u} Z_{iv} \le 1 \qquad i\in T, u\in U$$

$$\sum_{v\in W_u}(E_{i_1v} + TR_{W_u}\cdot Z_{i_1v}) + \sum_{\substack{i\in T\\ i_1<i<i_2}}\sum_{v\in W_u}(D_{iv} + TR_{W_u}\cdot Z_{iv})$$
$$\le \sum_{v\in W_u} S_{i_2v} + (H + TR_{W_u})\cdot(1 - \sum_{v\in W_u} Z_{i_2v}) \qquad i_1, i_2\in T, i_1<i_2, i_2>\underline{N_{W_u}}, u\in U$$

$$\sum_{v\in W} Z_{iv} \ge 1 \qquad i\in T$$

$$\sum_{v\in W_u} Z_{iv} \le \sum_{v\in W_u} Z_{(i-1)v} \qquad i\in T, i\neq 1, u\in U$$

$$\sum_{v\in W_u} Z_{iv} = 0 \qquad i\in T, i>\overline{N_{W_u}}, u\in U$$

$$\sum_{v\in W_u} Z_{iv} = 1 \qquad i\in T, i\le \underline{N_{W_u}}, u\in U$$

$$\sum_{v\in W_u}(E_{i_1v} + TR_{W_u}\cdot Z_{i_1v}) + \sum_{\substack{i\in T\\ i_1<i<i_2}}\sum_{v\in W_u}(D_{iv} + TR_{W_u}\cdot Z_{iv})$$
$$\le \sum_{v\in W_u} S_{i_2v} \qquad i_1, i_2\in T, i_1<i_2\le \underline{N_{W_u}}, u\in U$$

$$S_{iv}, D_{iv}, E_{iv} \ge 0 \qquad i\in T, v\in W$$

$$Z_{iv}\in\{0,1\} \qquad i\in T, v\in W$$

Cardinality bounds for units are defined similarly to the single-stage case. The minimum cardinality bound is obtained by considering the number of orders executed on units different than $u\in U$ that belong to the same stage l_u.

$$\overline{N_{W_u}} = \max\left\{\sum_{v\in W_u}\gamma_v \,\middle|\, \gamma_v\in\{0,1\}, \sum_{v\in W_u}(D_v + TR_{W_u})\cdot\gamma_v \le H + TR_{W_u}\right\} \qquad u\in U$$

$$\underline{N_{W_u}} = \max\left\{ |W_u^1|, |O| - \sum_{\substack{u'\in U\\ u'\neq u\\ l_{u'}=l_u}} \min\{n, \overline{N_{W_{u'}}}\}\right\} \qquad u\in U$$

Table 8: MOS computational results for Multi-Stage Batch Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
MSBSP5	2	6827.56	6827.56	0	0.98s	0.98s
	3	6828.76	$\underline{6828.76}$	58	1.66s	2.64
	4	6996.76	no solution	0	2.26s	$\underline{4.90s}$
	5	6996.76	no solution	0	3.23s	8.13s
MSBSP8	3	10985.61	10985.16	837	16.70s	16.70s
	4	11364.67	$\underline{10986.36}$	1149	59.87s	76.57s
	5	11364.67	no solution	500	30.34s	$\underline{106.91s}$
	6	11364.67	no solution	2200	156.14s	263.05s
	7	11364.67	no solution	1500	243.65s	506.70s
	8	11364.67	no solution	2200	331.18s	837.88s
MSBSP10	4	13627.41	13581.16	34340	+1000s	+1000s

The multi-stage batch scheduling problems are solved using the additive approach with the second stopping criterion ($n > \max_u \overline{N_u}$) which guarantees global optimality of the solution. The initial number of priority-slots is set to $n_0 = \left\lceil |O| \big/ \min_{l \in L} \left|\{u | u \in U_l\}\right| \right\rceil$, which is the minimum number of orders at least one unit has to process in each stage, more precisely in bottleneck stages. Computational results are given in Table 8. Similarly to the single-stage case, the results show that the first stopping criterion leads to significant reduction of CPU times compared to the second. The instance with 10 orders cannot be solved to global optimality although feasible solutions are obtained quickly using 4 priority-slots. The Branch & Bound search is stopped after 1000 seconds with a remaining 0.09% optimality gap. A 1% optimality gap can be achieved in 219.08 seconds.

8.2. MOS-SST Model

The MOS-SST model for the single-stage batch scheduling problem is derived from constraints (1)-(3), (10), (17), (20)-(22), and (27).

$$\textbf{Maximize} \quad \sum_{i \in T} \sum_{\substack{v \in W \\ v=(o,l,u)}} 0.2 \cdot l \cdot E_{iv}$$

Subject to

$$D_{iv} = D_v \cdot Z_{iv} \qquad i \in T, v \in W$$

$$E_{iv} \leq \overline{E_v} \cdot Z_{iv} \qquad i \in T, v \in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i \in T, v \in W$$

$$\sum_{i \in T} \sum_{v \in W_{ol}} Z_{iv} = 1 \qquad o \in O, l \in L$$

$$\underline{N_u} \leq \sum_{i \in T} \sum_{v \in W_u} Z_{iv} \leq \overline{N_u} \qquad u \in U$$

$$\sum_{i \in T} \sum_{v \in W_{o(l-1)}} E_{iv} \leq \sum_{i \in T} \sum_{v \in W_{ol}} S_{iv} \qquad o \in O, l \in L, l \neq 1$$

$$\sum_{v \in W_u} Z_{iv} \leq 1 \qquad i \in T, u \in U$$

$$\sum_{v \in W_u} (E_{i_1 v} + TR_{W_u} \cdot Z_{i_1 v}) + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W_u} (D_{iv} + TR_{W_u} \cdot Z_{iv})$$
$$\leq \sum_{v \in W_u} S_{i_2 v} + (H + TR_{W_u}) \cdot (1 - \sum_{v \in W_u} Z_{i_2 v}) \qquad i_1, i_2 \in T, i_1 < i_2, u \in U$$

$$t_{i-1} \leq t_i \qquad i \in T, i \neq 1$$

$$\sum_{v \in W_u} S_{iv} \leq t_i \qquad i \in T, u \in U$$

$$\sum_{v \in W_u} S_{iv} \geq t_i - H \cdot (1 - \sum_{v \in W'} Z_{iv}) \qquad i \in T, u \in U$$

$$\sum_{\substack{j \in T \\ j < i}} \sum_{v \in W_{o(l-1)}} Z_{jv} \geq \sum_{\substack{j \in T \\ j \leq i}} \sum_{v \in W_{ol}} Z_{jv} \qquad i \in T, o \in O, l \in L, l \neq 1$$

$$\sum_{v \in W} Z_{iv} \geq 1 \qquad i \in T$$

$$S_{iv}, D_{iv}, E_{iv} \geq 0 \qquad i \in T, v \in W$$

$$Z_{iv} \in \{0, 1\} \qquad i \in T, v \in W$$

$$t_i \in [0, H] \qquad i \in T$$

The additive approach is used to solve the model and the initial number of priority-slots is set to $n_0 = |L|$, the number of stages, as for each order stage processing operations must start at different dates. No problem instance was solved to global optimality, so we use a time limit of 1000 seconds. Computational results are given in Table 9. As for the single-stage case, the MOS-SST time representation is much less effective than the MOS time representation. For the instance with 10 orders the MOS-SST model returns a worse solution than the MOS model in 1000 seconds (13.31% optimality gap remaining).

8.3. MOS-FST Model

The MOS-FST model for the single-stage batch scheduling problem is derived from constraints (1)-(3), (10), (15), and (25).

Table 9: MOS-SST computational results for Multi-Stage Batch Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
MSBSP5	5	no solution			1.45s	1.45s
	6	6617.96	6161.40	21915	21.89s	23.34s
	7	6996.76	6448.62	285564	+1000s	+1023.34s
MSBSP8	5	no solution			1.81s	1.81s
	6	no solution			2.07s	3.88s
	7	11363.98	9979.40	21491	+1000s	+1003.88s
MSBSP10	5	no solution			2.19s	2.19s
	6	no solution			2.37s	4.56s
	7	no solution			3.36s	7.92s
	8	14255.76	12560.20	2172	+1000s	+1007.92s

$$\textbf{Maximize} \qquad \sum_{i\in T}\sum_{\substack{v\in W\\ v=(o,l,u)}} 0.2\cdot l\cdot E_{iv}$$

Subject to

$$D_{iv} = D_v\cdot Z_{iv} \qquad i\in T, v\in W$$

$$E_{iv} \le \overline{E_v}\cdot Z_{iv} \qquad i\in T, v\in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i\in T, v\in W$$

$$\sum_{i\in T}\sum_{v\in W_o} Z_{iv} = 1 \qquad o\in O$$

$$\underline{N_u} \le \sum_{i\in T}\sum_{v\in W_u} Z_{iv} \le \overline{N_u} \qquad u\in U$$

$$\sum_{i\in T}\sum_{v\in W_{o(l-1)}} E_{iv} \le \sum_{i\in T}\sum_{v\in W_{ol}} S_{iv} \qquad o\in O, l\in L, l\neq 1$$

$$S_{iv} = t_i\cdot Z_{iv} \qquad i\in T, v\in W$$

$$\sum_{v\in W_u}\left\{ Z_{iv} + \sum_{\substack{j\in T\\ i-\delta_u(v)<j<i}} Z_{jv}\right\} \le 1 \qquad i\in T, u\in U, \delta_u(v) = \left\lceil \frac{D_v + TR_u}{H/n}\right\rceil$$

$$S_{iv}, D_{iv}, E_{iv} \ge 0 \qquad i\in T, v\in W$$

$$Z_{iv}\in\{0,1\} \qquad i\in T, v\in W$$

$$t_i = \frac{i-1}{n}\cdot H \qquad i\in T$$

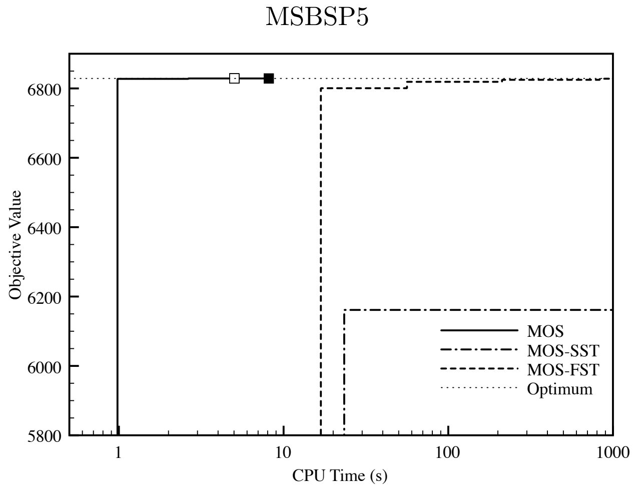
To solve this model, the multiplicative approach is used and the initial number of priority-slots is set to $n_0 = H/4 = 125$. We use a time limit of 1000s as a stopping criterion. Computational results are given in Table 10. As for the single-stage case, the MOS-FST time representation quickly finds near-optimal solutions in the first iterations. The instance with 10 orders was not solved due to memory limitations.

8.4. Models Comparison

Figure 10 shows how the objective value of the best incumbent varies over time. From this figure, it is clear that the MOS model is superior to the other models as it finds first feasible solutions very fast and is able to find optimal solutions in reasonable time. The MOS-FST model is significantly more expensive than previous single-stage cases as the scheduling

Table 10: MOS-FST computational results for Multi-Stage Batch Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
MSBSP5	125	6820.61	6800.60	0	16.84s	16.84s
	250	6826.21	6819.00	0	39.32s	56.16s
	500	6827.76	6824.80	499	155.83s	211.99s
	1000	6828.76	6828.00	509	619.69s	831.68s
	2000	6828.76	6828.10	0	+1000s	+1831.68s
MSBSP8	125	10951.49	10925.80	503	71.54s	71.54s
	250	10965.66	10956.60	31692	+1000s	+1071.54s



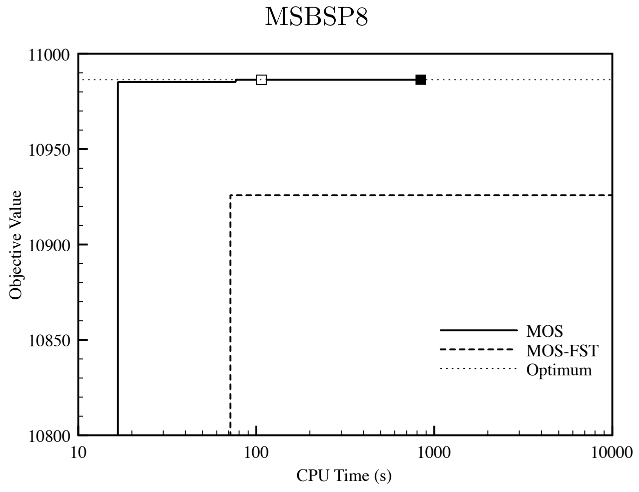


Figure 10: Comparison of the time representation for Multi-Stage Batch Scheduling Problems.

horizon is much longer (500 hours as opposed to 30 hours). Therefore, more priority-slots need to be postulated which makes the model size grow significantly.

9. Crude-Oil Operations Scheduling Problem

In this section, the different models are tested using the 4 refinery crude-oil scheduling problem studied in Mouret et al. (2009), noted COSP1, ...,COSP4. This problem has been widely studied from the optimization viewpoint since the work of Lee et al. (1996) and Shah (1996). It consists of crude-oil unloading from marine vessels to storage tanks, transfer and blending between tanks, and distillation of crude mixtures. The goal is to maximize profit and meet distillation demands for each type of crude blend (e.g. low sulfur or high sulfur blends), while satisfying unloading and transfer logistics constraints, inventory capacity limitations and property specifications for each blend. The logistics constraints involve non-overlapping constraints between crude-oil transfer operations:

(i) Only one berth is available at the docking station for vessel unloadings,

(ii) inlet and outlet transfers on tanks must not overlap,

(iii) a tank may charge only one CDU at a time,

(iv) a CDU can be charged by only one tank at a time,

(v) and CDUs must be operated continuously throughout the scheduling horizon.

Due to the logistics constraint (ii), inventory tracking in tanks can be performed by looking at the sequence of inlet and outlet operations only, and not at the sequence of start and end events of such operations. Indeed, for each tank, the scheduling solution can be decomposed into successions of inlet and outlet states during which one or several operations are performed. Therefore, it is only necessary to enforce capacity limitations at the transitions between these states. Figure 11

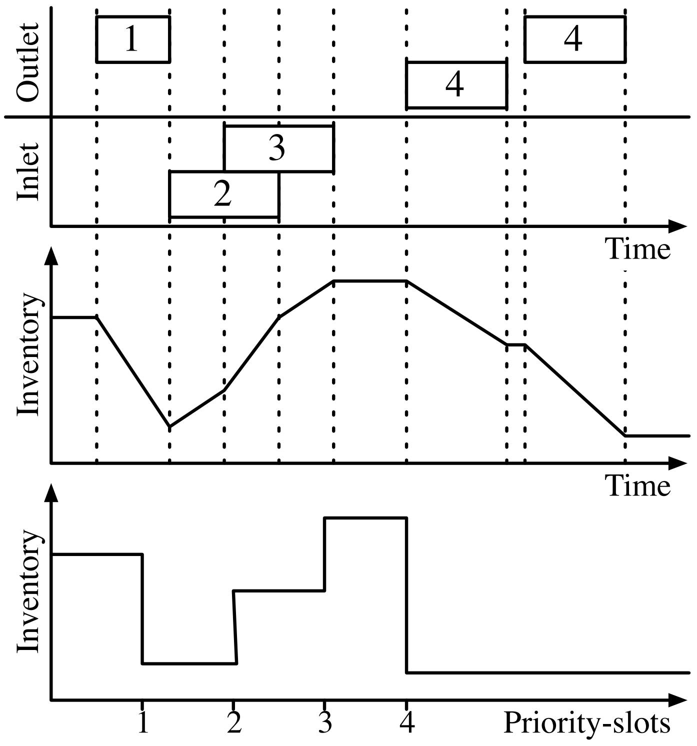


Figure 11: Example of tank schedule.

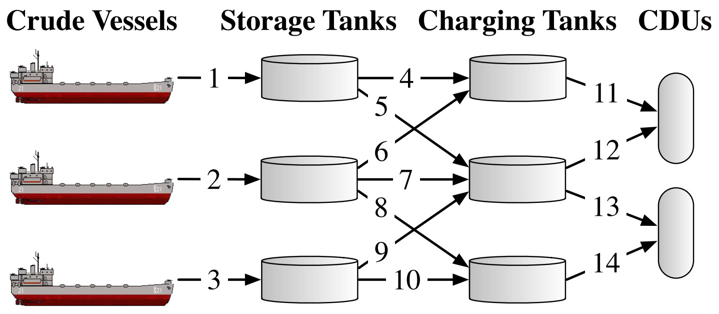
depicts an example of tank schedule with inlet and outlet states, time-based and priority-slot-based inventory profiles. Each transfer activity is represented by a horizontal bar labeled with the corresponding priority-slot. Vertical dashed lines represent transitions between outlet and inlet states. Under assumption (ii), it is sufficient to enforce tank capacity limitations just before transition slots as it corresponds to inventory upper and lower peaks. In practice, they are enforced just before all priority-slots.

It is well known that developing an efficient tool to solve this problem for any type of refinery is very hard as it is usually modeled as a large-scale MINLP and requires specific solution algorithms. In Mouret et al. (2009), the nonlinearities involved by transfer composition constraints were handled using a two step MILP-NLP procedure. In the first stage, an MILP relaxation obtained by removing the nonlinear composition constraints is solved. Although transfer stream compositions may be inconsistent with the upstream tank compositions, crude material balances are correctly enforced at this stage. Then, the discrete decisions (assignment variables Z_{iv}) of the optimal solution are fixed before solving the corresponding NLP (including the nonlinear constraints). Although global optimality is not proven, all instances were solved within a 4% optimality gap. We focus on the MILP solution as it is the most expensive and crucial step in the procedure.

Figure 12 displays the refinery system for problems 2 and 3. Each labelled arc corresponds to a transfer operation. Figure 13 displays the corresponding non-overlapping graph. It contains 4 maximal cliques of 3 operations: $\{1, 2, 3\}$, $\{5, 12, 13\}$, $\{7, 12, 13\}$, and $\{9, 12, 13\}$. In particular, maximal clique $\{5, 12, 13\}$ is due to non-overlapping constraints (ii) and (iii) which makes it non trivial to detect. Therefore, a generic maximal clique finding algorithm allows generating non-trivial strengthened non-overlapping constraints, which is the main objective of the non-overlapping graph representation.

The four time representations have been modeled using the same principles as in Mouret et al. (2009) for the SOS representation. The full mathematical models are presented in Appendix B.

Figure 12: Refinery crude-oil scheduling system for problems 2 and 3



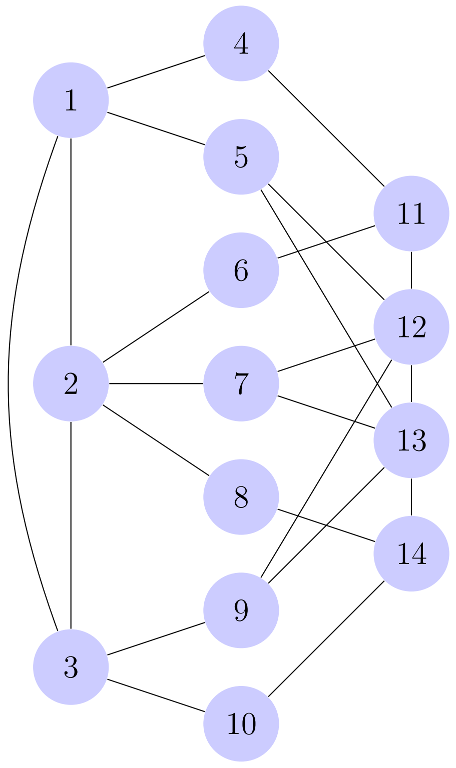


Figure 13: Non-overlapping graph for COSP2 and COSP3.

Table 11: MOS computational results for Crude-Oil Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
	1-4		infeas		3.08s	3.08s
COSP1	5	80.000	$\underline{79.750}$	48	1.16s	4.24s
	6	80.000	no solution	0	1.56s	$\underline{5.80s}$
	1-3		infeas		2.22s	2.22s
	4	103.000	90.000	0	1.05s	3.27s
COSP2	5	103.000	96.170	122	2.30s	5.57s
	6	103.000	$\underline{101.175}$	225	3.64s	9.21s
	7	103.000	no solution	335	8.61s	$\underline{17.82s}$
	1-2		infeas		1.43s	1.43s
	3	84.905	82.500	0	0.85s	2.28s
COSP3	4	100.000	84.500	16	1.26s	3.54s
	5	100.000	$\underline{87.400}$	63	1.74s	5.28s
	6	100.000	no solution	300	3.57s	$\underline{8.85s}$
	1-3		infeas		2.59s	2.59s
COSP4	4	132.585	$\underline{132.548}$	21	1.58s	4.17s
	5	132.585	no solution	0	1.72s	$\underline{5.89s}$

9.1. MOS Model

The MOS representation allows us to break symmetries in the model using the following generic constraint. It states that an operation v cannot be assigned to priority-slot i if no other non-overlapping operation is assigned to priority-slot $i-1$. Indeed, in this case operation v can be assigned to slot $i-1$ instead of slot i with no impact on the scheduling solution.

$$Z_{iv} \leq \sum_{\substack{v' \in W \\ NO_{vv'}=1}} Z_{(i-1)v'} \qquad\qquad i \in T, i > 1, v \in W \tag{34}$$

This constraint does not make any assumptions on the characteristics of the problem as opposed to symmetry-breaking constraint (29) which is specific to the case of batch scheduling problems.

The MOS model is solved with the additive approach using the second stop criterion ($\Delta \leq 0$). The initial number of priority-slots is set to $n_0 = 1$. Computational results are given in Table 11. All instances are solved within 20 seconds with a few priority-slots. The most difficult instance, that is COSP2, is the one requiring the most priority-slots alhough not the larger in size.

9.2. MOS-SST Model

The MOS-SST model is solved with the additive approach using the second stop criterion ($\Delta \leq 0$). The initial number of priority-slots is set to $n_0 = 1$. Computational results are given in Table 12. All instances are solved within 400 seconds with slightly more priority-slots than for the MOS model. The most difficult instance is still COSP2. It is interesting to note that for the solution obtained for this instance is not actually globally optimal, with a slightly lower objective value of 101.174 instead of 101.175 (see Table 11). The actual global optimal solution can be obtained with at least 10 priority-slots. It clearly shows that the second stopping criterion does not guarantee global optimality, although it is very efficient in practice.

9.3. MOS-FST Model

The MOS-FST model is solved with the multiplicative approach using the second stop criterion ($\Delta \leq 0$). The initial number of priority-slots is set to $n_0 = 4$. Computational results are given in Table 13. Only COSP1 is solved to global optimality although feasible solutions are obtained quickly. Within the 1000 second time limit, near-optimal solutions are obtained for instance COSP3 and COSP4, but the solution obtained for COSP3 shows a 4.8% gap with the best known solution. The MOS-FST discrete-time representation is not efficient at solving the crude-oil scheduling problem. Indeed,

Table 12: MOS-SST computational results for Crude-Oil Operations Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
COSP1	1-4		infeas		2.97s	2.97s
	5	80.000	79.722	144	1.25s	4.22s
	6	80.000	<u>79.750</u>	288	1.94s	6.16s
	7	80.000	no solution	968	6.30s	<u>12.46s</u>
COSP2	1-4		infeas		3.30s	3.30s
	5	103.000	90.000	327	4.59s	7.89s
	6	103.000	97.726	2316	33.51s	41.40s
	7	103.000	97.751	4672	70.26s	111.66s
	8	103.000	<u>101.174</u>	4545	88.45s	200.11s
	9	103.000	no solution	7007	185.29s	<u>385.40s</u>
COSP3	1-3		infeas		2.52s	2.52s
	4	100.000	82.500	48	1.43s	3.95s
	5	100.000	84.500	333	3.42s	7.37s
	6	100.000	87.000	931	14.83s	22.20s
	7	100.000	<u>87.400</u>	2101	38.02s	60.22s
	8	100.000	no solution	3900	99.03s	<u>159.25s</u>
COSP4	1-5		infeas		24.84s	24.84s
	6	132.585	<u>132.548</u>	1293	28.62s	53.46s
	7	132.585	no solution	500	20.74s	<u>74.20s</u>

Table 13: MOS-FST computational results for Crude-Oil Operations Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
COSP1	4	80.000	infeas	0	0.87s	0.87s
	8	80.000	<u>79.750</u>	74	1.56s	1.56s
	16	80.000	no solution	100	15.17s	<u>17.60s</u>
COSP2	4	98.000	infeas	0	0.90s	0.90s
	8	103.000	90.000	2114	28.76s	29.66s
	16	103.000	96.250	11852	+1000s	+1029.66s
COSP3	4	84.929	82.500	0	0.93s	0.93s
	8	99.120	84.250	22	2.55s	3.48s
	16	100.000	87.156	563	29.97s	33.45s
	32	100.000	no solution	300	+1000s	+1033.45s
COSP4	4	132.585	infeas	0	1.20s	1.20s
	8	132.585	132.266	574	12.82s	14.02s
	16	132.585	132.362	2769	376.84s	390.86s
	32	132.585	no solution	0	+1000s	+1390.86s

as variable processing times are used, constraint (5) does not help strengthening the model. Instead, if one wishes to use a discrete-time approach, it would be preferable to consider the work of Lee et al. (1996). Indeed, in their formulation, each operation that is executed over several consecutive time intervals is actually split into several smaller operations, one for each time interval.

9.4. SOS Model

As mentioned in Section 5, both cliques and bicliques of G_{NO} can be used to generate non-overlapping constraints. Table 14 displays all maximal cliques and all maximal bicliques that are not derived from cliques for instances COSP2 and COSP3. They are 15 maximal cliques and 15 maximal bicliques which corresponds to $15 + 2 \cdot 15 = 45$ constraints such as (19) and (26). There is a clear trade-off between the tightness of the LP relaxation and the size of the model. Therefore, it is important to carefully select the cliques and bicliques that are used to enforce non-overlapping constraints. We introduce 3 different selection heuristics. The first selection strategy, denoted by a, consists in selecting all maximal cliques only, leading to 15 constraints (19). The second selection strategy, denoted by b, consists in deriving cliques and bicliques directly from the non-overlapping constraint definitions. The third selection strategy, denoted by c, consists in improving selection

Table 14: Maximal cliques and bicliques for COSP2 and COSP3

Nb of vertices	Maximal cliques
2	$\{1,4\}$, $\{1,5\}$, $\{2,6\}$, $\{2,7\}$, $\{2,8\}$, $\{3,9\}$, $\{3,10\}$, $\{4,11\}$, $\{6,11\}$, $\{8,14\}$, $\{10,14\}$, $\{11,12\}$, $\{13,14\}$
3	$\{1,2,3\}$, $\{5,12,13\}$, $\{7,12,13\}$, $\{9,12,13\}$

Nb of vertices	Maximal bicliques
3	$(\{4\};\{1,4,11\})$, $(\{6\};\{2,6,11\})$, $(\{8\};\{2,8,14\})$, $(\{10\};\{3,10,14\})$
4	$(\{5\};\{1,5,12,13\})$, $(\{7\};\{2,7,12,13\})$, $(\{9\};\{3,9,12,13\})$, $(\{11\};\{4,6,11,12\})$, $(\{14\};\{8,10,13,14\})$
5	$(\{1\};\{1,2,3,4,5\})$, $(\{3\};\{1,2,3,9,10\})$, $(\{5,7,9,12,13\};\{12,13\})$
6	$(\{2\};\{1,2,3,6,7,8\})$, $(\{12\};\{5,7,9,11,12,13\})$, $(\{13\};\{5,7,9,12,13,14\})$

Table 15: Cliques and bicliques used in selections b and c

Constraint	No.	Selection a	Selection b	Selection c
(i)	1	$\{1,2,3\}$	$\{1,2,3\}$	implied by 2, 3, 4
(ii)	2	$\{1,4\}$, $\{1,5\}$	$(\{1\};\{4,5\})$	$(\{1\};\{1,2,3,4,5\})$
	3	$\{2,6\}$, $\{2,7\}$, $\{2,8\}$	$(\{2\};\{6,7,8\})$	$(\{2\};\{1,2,3,6,7,8\})$
	4	$\{3,9\}$, $\{3,10\}$	$(\{3\};\{9,10\})$	$(\{3\};\{1,2,3,9,10\})$
	5	$\{4,11\}$, $\{6,11\}$	$(\{4,6\};\{11\})$	$(\{4,6,11,12\};\{11\})$
	6	$\{5,12,13\}$, $\{7,12,13\}$, $\{9,12,13\}$	$(\{5,7,9\};\{12,13\})$	$(\{5,7,9,12,13\};\{12,13\})$
	7	$\{8,14\}$, $\{10,14\}$	$(\{8,10\};\{14\})$	$(\{8,10,13,14\};\{14\})$
(iii)	8	implied by 6	$\{12,13\}$	implied by 6
(iv)	9	$\{11,12\}$	$\{11,12\}$	implied by 5
	10	$\{13,14\}$	$\{13,14\}$	implied by 7

b by extending it to maximal cliques and bicliques and removing unecessary elements from the selection. Table 15 shows selections a, b and c. For example, non-overlapping constraint (ii) for the first storage tank is intuitively represented by the clique $(\{1\};\{4,5\})$ in selection b but it is extended to the maximal biclique $(\{1\};\{1,2,3,4,5\})$ in selection c. Also, non-overlapping constraint (i) is represented by the maximal clique $\{1,2,3\}$ in selection b but is removed in selection c as it is implied by bicliques 2, 3, and 4. Therefore, selection b leads to 16 non-overlapping constraints while selection c leads to 12 non-overlapping constraints only.

Similarly to the MOS representation, it is possible to derive efficient symmetry-breaking constraints for SOS models. Mouret et al. (2009) described a method for restricting the set of feasible sequences of operations. They used a deterministic finite automaton (DFA) to express a sequencing rule which was then included in the model as a network flow problem. This symmetry-breaking approach is problem specific as it exploits in details its scheduling properties.

The SOS model is solved using the direct approach. The number of priority-slots is set to its maximum value as determined by Mouret et al. (2009). The authors mentioned that this approach does not guarantee global optimality of the solution, although, in practice, the returned solution is always globally optimal. Computational results are given in Table 16 for each clique/biclique selection heuristic. The behaviour of this time representation is quite different than for the MOS and MOS-SST representations as instance COSP4 is now the most difficult, and instance COSP2 is quite easy to solve. Indeed, COSP4 is solved with the largest number of priority-slots as it is the largest instance in terms of operations and resources, which makes it the hardest instance. The results obtained for selection a shows that using only maximal cliques is not the most efficient in the SOS model. It is rather preferable to combine it with bicliques as in selections b and c. Additionally, it is clear that the selection improvements in selection c lead to a significant decrease in CPU time for instances COSP3 and COSP4. This is due to the fact that non-overlapping constraints are strengthened and model size is reduced.

Table 16: SOS computational results for Crude-Oil Operations Scheduling Problems

Pb	n	LP	MILP	Nb of nodes	CPU
COSP1[a]	13	80.000	79.750	18	5.88s
COSP1[b]	13	80.000	79.750	19	4.45s
COSP1[c]	13	80.000	79.750	21	4.92s
COSP2[a]	21	103.000	101.175	36	120.42s
COSP2[b]	21	103.000	101.175	19	55.57s
COSP2[c]	21	103.000	101.175	25	60.50s
COSP3[a]	21	100.000	87.400	28	191.47s
COSP3[b]	21	100.000	87.400	33	97.70s
COSP3[c]	21	100.000	87.400	31	64.46s
COSP4[a]	26	132.585	132.548	16	606.86s
COSP4[b]	26	132.585	132.548	37	574.95s
COSP4[c]	26	132.585	132.548	32	308.43s

Table 17: Size of MOS, MOS-SST, MOS-FST, and SOS models

		n	Binary Vars	Continuous Vars	Constraints
COSP1	MOS	5	40	496	1086
	MOS-SST	6	48	601	1403
	MOS-FST	8	64	793	1804
	SOS	13	104	2848	2333
COSP2	MOS	6	84	1303	2620
	MOS-SST	8	112	1745	3758
	MOS-FST	16	224	3473	8051
	SOS	21	294	13084	7835
COSP3	MOS	5	70	1211	2302
	MOS-SST	7	98	1702	3431
	MOS-FST	16	224	3873	8439
	SOS	21	294	13609	7909
COSP4	MOS	4	76	1489	2806
	MOS-SST	6	114	2239	4338
	MOS-FST	16	304	5953	12269
	SOS	26	494	28445	14351

9.5. Models Comparison

Figure 14 shows how the objective value of the best incumbent varies over time. Results for the SOS model are presented for clique/biclique selection heuristic *c*. As for the previous batch scheduling problems the MOS model shows to be superior. However, the MOS-SST model behaves better than he MOS-FST model for these problems due to non-constant processing times. The SOS model performs better than the MOS-SST model for instance COSP2, worse for instance COSP4, and similarly for the other instances. From these results, it seems that the MOS-SST model scales better with the size of the problem than the SOS model as it requires fewer priority-slots. Finally, in Table 17 we present the model sizes of the MILPs corresponding to all time representations using the number of priority-slots leading to the best solution. It is clear that the MOS time representation leads to the most compact models which partly explains its efficiency. The MOS-FST time representation is much larger due to the higher number of priority-slots required. Also, it is interesting to note that the SOS time representation requires many more continuous variables. These variables are introduced to represent the network flow used to break symmetries. Overall, it is also interesting to note that 4 very different models all provide the same LP relaxation for the problem. Indeed, the objective function is purely economic and not directly linked to scheduling issues, as opposed to the previous batch scheduling examples. In the case of crude-oil operations scheduling, solving the LP relaxation corresponds to generating a solution that only satisfies overall material balances, distillation demands and specifications, and capacity limits at the end of the scheduling horizon.

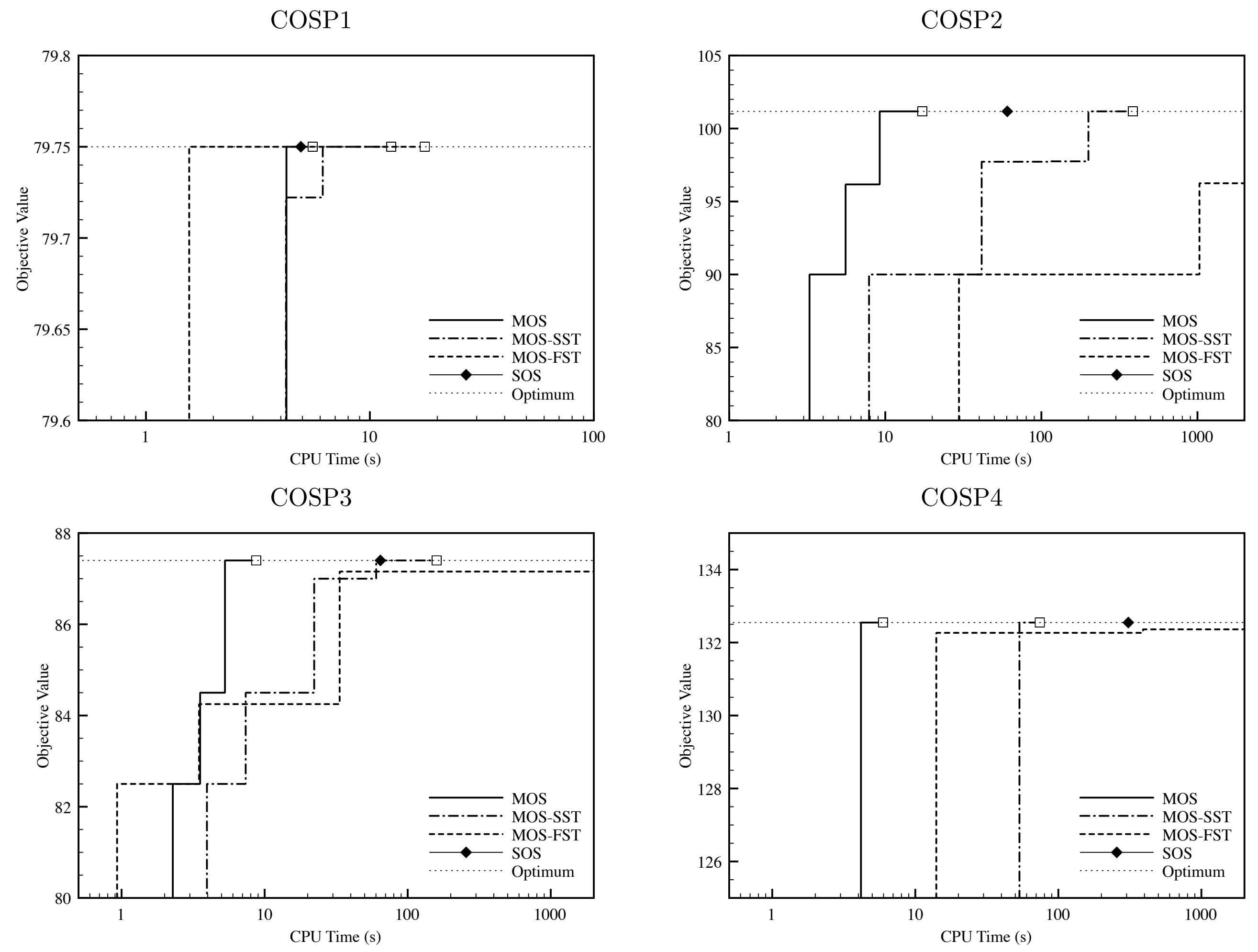


Figure 14: Comparison of the time representation for Crude Oil Operations Scheduling Problems.

10. Conclusion

In this paper, we presented four different time representations that in different forms have been previously introduced to solve chemical scheduling problems. Using the common concept of priority-slot, it is shown that it is possible to derive relationship results between these time representations. Additionally, generic scheduling constraints are developed as well as specific solution algorithms for each model. We apply these models to batch scheduling of single and multi-stage plants, and to crude-oil operations scheduling.

Intuitively, the representation that requires the least priority-slots is the most computationally effective. In practice, the MOS time representation proves to be superior to the other time representations on the three types of problem tested due to the smaller number of priority-slots and symmetry-breaking constraints used. The discrete-time MOS-FST time representation behaves well on problems with constant processing times while the MOS-SST time representation performs better with variable processing times. Although it requires a larger number of priority-slots, the SOS time representation is comparable to the MOS-SST time representation for the crude-oil operations scheduling problems. In particular, the SOS model is efficiently solved using DFA-based symmetry-breaking constraints (see Mouret et al., 2009).

A natural extension of this work is to adapt the time representations and corresponding mathematical models to more complex problems that require precise tracking of operation start and end events. Besides, instead of directly solving the models using an MILP solver, multi-stage algorithms can be developed in order to decompose the different decision levels, such as *selection* and *sequencing* of operations.

11. Acknowledgment

The authors are grateful to Total Raffinage & Marketing for financial support of this project.

Appendix A. On Tightness of Strengthened Constraints

We present mathematical results on the tightness of strengthened constraints.

- Constraint (17) is at least as tight as constraint (4). Indeed, assume constraint (17) is satisfied for W' such that $v_1, v_2 \in W'$:

$$Z_{iv_1} + Z_{iv_2} \leq \sum_{v \in W'} Z_{iv} \leq 1$$

- Constraint (6) is at least as tight as constraints (5). Indeed, assume constraint (6) is satisfied for $v_1, v_2 \in W$:

$$\begin{aligned} E_{i_1 v_1} &\leq E_{i_1 v_1} + E_{i_1 v_2} \\ &\leq S_{i_2 v_1} + S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_1} - Z_{i_2 v_2}) \\ &\leq S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_2}) + S_{i_2 v_1} - H \cdot Z_{i_2 v_1} \\ &\leq S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_2}) \end{aligned}$$

- Constraint (18) is at least as tight as constraint (6). Indeed, assume constraint (18) is satisfied for W' such that

$v_1, v_2 \in W'$:

$$\begin{aligned} E_{i_1v_1} + E_{i_1v_2} &\leq \sum_{v\in W'} E_{i_1v} \\ &\leq \sum_{v\in W'} S_{i_2v} + H\cdot\left(1 - \sum_{v\in W'} Z_{i_2v}\right) \\ &\leq S_{i_2v_1} + S_{i_2v_2} + H\cdot(1 - Z_{i_2v_1} - Z_{i_2v_2}) + \sum_{v\in W'\setminus\{v_1,v_2\}} (S_{i_2v} - H\cdot Z_{i_2v}) \\ &\leq S_{i_2v_1} + S_{i_2v_2} + H\cdot(1 - Z_{i_2v_1} - Z_{i_2v_2}) \end{aligned}$$

- Constraint (19) is at least as tight as constraint (18). Indeed, assume constraint (19) is satisfied for W':

$$\sum_{v\in W'} E_{i_1v} \leq \sum_{v\in W'} E_{i_1v} + \sum_{\substack{i\in T\\ i_1<i<i_2}} \sum_{v\in W'} D_{iv} \leq \sum_{v\in W'} S_{i_2v} + H\cdot(1 - \sum_{v\in W'} Z_{i_2v})$$

- Constraint (21a) is at least as tight as constraint (13a). Indeed, assume constraint (21a) is satisfied for W' such that $v' \in W'$:

$$S_{iv} \leq \sum_{v'\in W'} S_{iv'} \leq t_i$$

- Constraint (21b) is at least as tight as constraint (13b). Indeed, assume constraint (21b) is satisfied for W' such that $v \in W'$:

$$\begin{aligned} S_{iv} &\geq S_{iv} + \sum_{v'\in W'\setminus\{v\}} (S_{iv'} - H\cdot Z_{iv'}) \\ &\geq \sum_{v'\in W'} S_{iv'} - H\cdot \sum_{v'\in W'\setminus\{v\}} Z_{iv'} \\ &\geq t_i - H\cdot(1 - \sum_{v'\in W'} Z_{iv'}) - H\cdot \sum_{v'\in W'\setminus\{v\}} Z_{iv'} \\ &\geq t_i - H\cdot(1 - Z_{iv}) \end{aligned}$$

- Constraint (26) is at least as tight as constraint (5). Indeed, assume constraint (26) is satisfied for sets of operations W_1 and W_2 such that $v_1 \in W_1$ and $v_2 \in W_2$:

$$\begin{aligned} E_{i_1v_1} &\leq \sum_{v\in W_1} E_{i_1v} \\ &\leq \sum_{v\in W_2} S_{i_2v} + H\cdot(1 - \sum_{v\in W_2} Z_{i_2v}) \\ &\leq S_{i_2v_2} + H\cdot(1 - Z_{i_2v_2}) + \sum_{v\in W_2\setminus\{v_2\}} (S_{i_2v} - H\cdot Z_{i_2v}) \\ &\leq S_{i_2v_2} + H\cdot(1 - Z_{i_2v_2}) \end{aligned}$$

Appendix B. Mathematical Models for Crude-Oil Operations Scheduling Problems

We present the four mathematical models for crude-oil operations scheduling problems corresponding to each time representation. They make use of the following additional sets and parameters.

- $W_U \subset W$ is the set of unloading operations ($W_U = \{1, 2, 3\}$ for COSP2)

- $W_T \subset W$ is the set of tank-to-tank transfer operations ($W_T = \{4, \ldots, 10\}$ for COSP2)
- $W_D \subset W$ is the set of distillation operations ($W_D = \{11, \ldots, 14\}$ for COSP2)
- R is the set of resources (i.e. tanks, units): $R = R_V \cup R_S \cup R_C \cup R_D$
- $R_V \subset R$ is the set of vessels
- $R_S \subset R$ is the set of storage tanks
- $R_C \subset R$ is the set of charging tanks
- $R_D \subset R$ is the set of distillation units
- $I_r \subset W$ is the set of inlet transfer operations on resource r
- $O_r \subset W$ is the set of outlet transfer operations on resource r
- C is the set of products (i.e. crudes)
- K is the set of product properties (e.g. crude sulfur concentration)
- $[\underline{V_v^t}, \overline{V_v^t}]$ are bounds on the total volume transferred during transfer operation v ; in all instances, $\underline{V_v^t} = 0$ for all operations except unloadings for which $\underline{V_v^t} = \overline{V_v^t}$ is the volume of crude in the marine vessel
- $[\underline{N_D}, \overline{N_D}]$ are the bounds on the number of distillations
- $[\underline{FR_v}, \overline{FR_v}]$ are flowrate limitations for transfer operation v
- $[\underline{x_{vk}}, \overline{x_{vk}}]$ are the limits of property k of the blended products transferred during operation v
- x_{ck} is the value of the property k of crude c
- $[\underline{L_r^t}, \overline{L_r^t}]$ are the capacity limits of tank r
- $[\underline{D_r}, \overline{D_r}]$ are the bounds of the demand on products to be transferred out of the charging tank r during the scheduling horizon
- G_c is the gross margin of crude c

Furthermore, the following additional variables are introduced.

- **Operation variables** $V_{iv}^t \geq 0$ and $V_{ivc} \geq 0 \qquad i \in T, v \in W, c \in C$

  V_{iv}^t is the total volume of crude transferred during operation v if it is assigned to priority-slot i, $V_{iv}^t = 0$ otherwise.

  V_{ivc} is the volume of crude c transferred during operation v if it is assigned to priority-slot i, $V_{ivc} = 0$ otherwise.

- **Resource variables** L_{ir}^t and $L_{irc} \qquad i \in T, r \in R, c \in C$

  L_{ir}^t is the total *accumulated* level of crude in tank $r \in R_S \cup R_C$ before the operation assigned to priority-slot i.

  L_{irc} is the *accumulated* level of crude c in tank $r \in R_S \cup R_C$ before the operation assigned to priority-slot i.

In all models, the following variable (B.1), operation (B.2) and resource (B.3) constraints are used.

$$V_{iv}^t \leq \overline{V_v^t} \cdot Z_{iv} \qquad i \in T, v \in W \tag{B.1a}$$

$$V_{iv}^t \geq \underline{V_v^t} \cdot Z_{iv} \qquad i \in T, v \in W \tag{B.1b}$$

$$V_{iv}^t = \sum_{c \in C} V_{ivc} \qquad i \in T, v \in W \tag{B.1c}$$

$$L_{irc} = L_{0rc} + \sum_{j \in T, j<i} \sum_{v \in I_r} V_{ivc} - \sum_{j \in T, j<i} \sum_{v \in O_r} V_{ivc} \qquad i \in T, r \in R, c \in C \tag{B.1d}$$

$$L_{ir}^t = \sum_{c \in C} L_{irc} \qquad i \in T, r \in R \tag{B.1e}$$

$$\underline{FR_v} \cdot D_{iv} \leq V_{iv}^t \leq \overline{FR_v} \cdot D_{iv} \qquad i \in T, v \in W \tag{B.2a}$$

$$\underline{x_{vk}} \cdot V_{iv}^t \leq \sum_{c \in C} x_{ck} V_{ivc} \leq \overline{x_{vk}} \cdot V_{iv}^t \qquad i \in T, v \in W, k \in K \tag{B.2b}$$

$$\underline{L_r^t} \leq L_{ir}^t \leq \overline{L_r^t} \qquad i \in T, r \in R_S \cup R_C \tag{B.3a}$$

$$0 \leq L_{irc} \leq \overline{L_r^t} \qquad i \in T, r \in R_S \cup R_C, c \in C \tag{B.3b}$$

$$\underline{L_r^t} \leq L_{0r}^t + \sum_{i \in T} \sum_{v \in I_r} V_{iv}^t - \sum_{i \in T} \sum_{v \in O_r} V_{iv}^t \leq \overline{L_r^t} \qquad r \in R_S \cup R_C \tag{B.3c}$$

$$0 \leq L_{0rc} + \sum_{i \in T} \sum_{v \in I_r} V_{ivc} - \sum_{i \in T} \sum_{v \in O_r} V_{ivc} \leq \overline{L_r^t} \qquad r \in R_S \cup R_C, c \in C \tag{B.3d}$$

$$\underline{D_r} \leq \sum_{i \in T} \sum_{v \in O_r} V_{iv}^t \leq \overline{D_r} \qquad r \in R_C \tag{B.3e}$$

$$\textbf{Maximize} \qquad \sum_{i\in T}\sum_{r\in R_D}\sum_{v\in I_r}\sum_{c\in C} G_c \cdot V_{ivc}$$

Subject to

$$S_{iv} \geq \underline{S_v} \cdot Z_{iv} \qquad i\in T, v\in W_U$$

$$E_{iv} \leq H \cdot Z_{iv} \qquad i\in T, v\in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i\in T, v\in W$$

$$\sum_{i\in T}\sum_{v\in O_r} Z_{iv} = 1 \qquad r\in R_V$$

$$\underline{N_D} \leq \sum_{i\in T}\sum_{v\in W_D} Z_{iv} \leq \overline{N_D}$$

$$\sum_{i\in T}\sum_{v\in O_{r_1}} E_{iv} \leq \sum_{i\in T}\sum_{v\in O_{r_2}} S_{iv} \qquad r_1, r_2 \in R_V, r_1 < r_2$$

$$\sum_{\substack{j\in T\\ j<i}}\sum_{v\in O_{r_1}} Z_{jv} \geq \sum_{\substack{j\in T\\ j\leq i}}\sum_{v\in O_{r_2}} Z_{jv} \qquad i\in T, r_1, r_2\in R_V, r_1 < r_2$$

$$\sum_{i\in T}\sum_{v\in I_r} D_{iv} = H \qquad r\in R_D$$

Assignment constraint (17)

Non-overlapping constraint (19)

$$Z_{iv} \leq \sum_{\substack{v'\in W\\ NO_{vv'}=1}} Z_{(i-1)v'} \qquad i\in T, i\neq 1, v\in W$$

$$\sum_{v\in W} Z_{iv} \geq 1 \qquad i\in T$$

Variable constraints (B.1)

Operation constraints (B.2)

Resource constraints (B.3)

$$S_{iv}, D_{iv}, E_{iv}, V_{iv}^t \geq 0 \qquad i\in T, v\in W$$

$$V_{ivc} \geq 0 \qquad i\in T, v\in W, c\in C$$

$$L_{ir}^t \geq 0 \qquad i\in T, r\in R_S\cup R_C$$

$$L_{irc} \geq 0 \qquad i\in T, r\in R_S\cup R_C, c\in C$$

$$Z_{iv} \in \{0,1\} \qquad i\in T, v\in W$$

$$\textbf{Maximize} \quad \sum_{i \in T} \sum_{r \in R_D} \sum_{v \in I_r} \sum_{c \in C} G_c \cdot V_{ivc}$$

Subject to

$$S_{iv} \geq \underline{S_v} \cdot Z_{iv} \qquad i \in T, v \in W_U$$

$$E_{iv} \leq H \cdot Z_{iv} \qquad i \in T, v \in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i \in T, v \in W$$

$$\sum_{i \in T} \sum_{v \in O_r} Z_{iv} = 1 \qquad r \in R_V$$

$$\underline{N_D} \leq \sum_{i \in T} \sum_{v \in W_D} Z_{iv} \leq \overline{N_D}$$

$$\sum_{i \in T} \sum_{v \in O_{r_1}} E_{iv} \leq \sum_{i \in T} \sum_{v \in O_{r_2}} S_{iv} \qquad r_1, r_2 \in R_V, r_1 < r_2$$

$$\sum_{\substack{j \in T \\ j < i}} \sum_{v \in O_{r_1}} Z_{jv} \geq \sum_{\substack{j \in T \\ j \leq i}} \sum_{v \in O_{r_2}} Z_{jv} \qquad i \in T, r_1, r_2 \in R_V, r_1 < r_2$$

$$\sum_{i \in T} \sum_{v \in I_r} D_{iv} = H \qquad r \in R_D$$

Assignment constraint (17)

Non-overlapping constraint (19)

$$t_{i-1} \leq t_i \qquad i \in T, i \neq 1$$

$$\sum_{v \in W'} S_{iv} \leq t_i \qquad i \in T, W' \in clique(G_{NO})$$

$$\sum_{v \in W'} S_{iv} \geq t_i - H \cdot (1 - \sum_{v \in W'} Z_{iv}) \qquad i \in T, W' \in clique(G_{NO})$$

$$\sum_{v \in W} Z_{iv} \geq 1 \qquad i \in T$$

Variable constraints (B.1)

Operation constraints (B.2)

Resource constraints (B.3)

$$S_{iv}, D_{iv}, E_{iv}, V_{iv}^t \geq 0 \qquad i \in T, v \in W$$

$$V_{ivc} \geq 0 \qquad i \in T, v \in W, c \in C$$

$$L_{ir}^t \geq 0 \qquad i \in T, r \in R_S \cup R_C$$

$$L_{irc} \geq 0 \qquad i \in T, r \in R_S \cup R_C, c \in C$$

$$Z_{iv} \in \{0, 1\} \qquad i \in T, v \in W$$

$$t_i \in [0, H] \qquad i \in T$$

$$\textbf{Maximize} \qquad \sum_{i\in T}\sum_{r\in R_D}\sum_{v\in I_r}\sum_{c\in C} G_c \cdot V_{ivc}$$

Subject to

$$S_{iv} \geq \underline{S_v} \cdot Z_{iv} \qquad i \in T, v \in W_U$$

$$E_{iv} \leq H \cdot Z_{iv} \qquad i \in T, v \in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i \in T, v \in W$$

$$\sum_{i\in T}\sum_{v\in O_r} Z_{iv} = 1 \qquad r \in R_V$$

$$\underline{N_D} \leq \sum_{i\in T}\sum_{v\in W_D} Z_{iv} \leq \overline{N_D}$$

$$\sum_{i\in T}\sum_{v\in O_{r_1}} E_{iv} \leq \sum_{i\in T}\sum_{v\in O_{r_2}} S_{iv} \qquad r_1, r_2 \in R_V, r_1 < r_2$$

$$\sum_{i\in T}\sum_{v\in I_r} D_{iv} = H \qquad r \in R_D$$

Assignment constraint (17)

Non-overlapping constraint (19)

$$S_{iv} = t_i \cdot Z_{iv} \qquad i \in T, v \in W$$

$$\sum_{v\in W} Z_{iv} \geq 1 \qquad i \in T$$

Variable constraints (B.1)

Operation constraints (B.2)

Resource constraints (B.3)

$$S_{iv}, D_{iv}, E_{iv}, V^t_{iv} \geq 0 \qquad i \in T, v \in W$$

$$V_{ivc} \geq 0 \qquad i \in T, v \in W, c \in C$$

$$L^t_{ir} \geq 0 \qquad i \in T, r \in R_S \cup R_C$$

$$L_{irc} \geq 0 \qquad i \in T, r \in R_S \cup R_C, c \in C$$

$$Z_{iv} \in \{0, 1\} \qquad i \in T, v \in W$$

$$t_i = \frac{i-1}{n} \cdot H \qquad i \in T$$

$$\textbf{Maximize} \qquad \sum_{i\in T}\sum_{r\in R_D}\sum_{v\in I_r}\sum_{c\in C} G_c \cdot V_{ivc}$$

Subject to

$$S_{iv} \geq \underline{S_v} \cdot Z_{iv} \qquad i \in T, v \in W_U$$

$$E_{iv} \leq H \cdot Z_{iv} \qquad i \in T, v \in W$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i \in T, v \in W$$

$$\sum_{i\in T}\sum_{v\in O_r} Z_{iv} = 1 \qquad r \in R_V$$

$$\underline{N_D} \leq \sum_{i\in T}\sum_{v\in W_D} Z_{iv} \leq \overline{N_D}$$

$$\sum_{i\in T}\sum_{v\in O_{r_1}} E_{iv} \leq \sum_{i\in T}\sum_{v\in O_{r_2}} S_{iv} \qquad r_1, r_2 \in R_V, r_1 < r_2$$

$$\sum_{\substack{j\in T\\ j<i}}\sum_{v\in O_{r_1}} Z_{jv} \geq \sum_{\substack{j\in T\\ j\leq i}}\sum_{v\in O_{r_2}} Z_{jv} \qquad i \in T, r_1, r_2 \in R_V, r_1 < r_2$$

$$\sum_{i\in T}\sum_{v\in I_r} D_{iv} = H \qquad r \in R_D$$

Assignment constraint (17)

Non-overlapping constraints (19) and (26)

Symmetry-breaking constraints (Mouret et al., 2009)

$$\sum_{v\in W} Z_{iv} \geq 1 \qquad i \in T$$

Variable constraints (B.1)

Operation constraints (B.2)

Resource constraints (B.3)

$$S_{iv}, D_{iv}, E_{iv}, V_{iv}^t \geq 0 \qquad i \in T, v \in W$$

$$V_{ivc} \geq 0 \qquad i \in T, v \in W, c \in C$$

$$L_{ir}^t \geq 0 \qquad i \in T, r \in R_S \cup R_C$$

$$L_{irc} \geq 0 \qquad i \in T, r \in R_S \cup R_C, c \in C$$

$$Z_{iv} \in \{0, 1\} \qquad i \in T, v \in W$$

References

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job shop scheduling. Management Science 34 (3), 391–401.

Floudas, C. A., Lin, X., 2004. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Computers and Chemical Engineering 28, 2109–2129.

Ierapetritou, M. G., Floudas, C. A., 1998. Effective continuous-time formulation for short-term scheduling. 1. multipurpose batch processes. Industrial and Engineering Chemistry Research 37 (11), 4341–4359.

Janak, S. L., Lin, X., Floudas, C. A., 2004. Enhance continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: Resource consraints and mixed storage policies. Industrial and Engineering Chemistry Research 43 (10), 2516–2533.

Kallrath, J., 2002. Planning and scheduling in the process industry. OR Spectrum 24 (3), 219–250.

Kondili, E., Pantelides, C. C., Sargent, R. W. H., 1993. A general algorithm for short-term scheduling of batch operations. i: Milp formulation. Computers and Chemical Engineering 17 (2), 211–227.

Lee, H., Pinto, J. M., Grossmann, I. E., Park, S., 1996. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. Industrial and Engineering Chemistry Research 35 (5), 1630–1641.

Maravelias, C. T., Grossmann, I. E., 2003. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Industrial and Engineering Chemistry Research 42 (13), 3056–3074.

Mouret, S., Grossmann, I. E., Pestiaux, P., 2009. A novel priority-slot based continuous-time formulation for crude-oil scheduling problems. Industrial and Engineering Chemistry Research 48 (18), 8515–8528.

Nemhauser, G. L., Wolsey, L. A., 1999. Integer and Combinatorial Optimization. Wiley-Interscience.

Pantelides, C. C., 1994. Unified frameworks for optimal process planning and scheduling. In: Rippin, D., Hale, J., Davis, J. (Eds.), Proceedings of the Second International Conference on Foundations of Computer-Aided Process Operations. pp. 253–274.

Pinto, J. M., Grossmann, I. E., 1995. A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. Industrial and Engineering Chemistry Research 34 (9), 3037–3051.

Rossi, F., van Beek, P., Walsh, T. (Eds.), 2006. Handbook of Constraint Programming. Elsevier.

Schilling, G., Pantelides, C. C., 1996. A simple continuous-time process scheduling formulation and a novel solution algorithm. Computers and Chemical Engineering 20 (Supplement 2), S1221–S1226.

Shah, N., 1996. Mathematical programming techniques for crude oil scheduling. Computers and Chemical Engineering 20, 1227–1232.

Zhang, X., Sargent, R. W. H., 1996. The optimal operation of mixed production facilities: a general formulation and some approaches for the solution. Computers and Chemical Engineering 20 (6-7), 897–904.