Exact and heuristic methods for an aeronautical assembly line time-constrained scheduling problem with multiple modes and a resource leveling objective

Tamara Sanchidrián, Tom Portoleau, Christian Artigues, Alvaro García Sánchez, Miguel Ortega Mier, Pierre Lopez

Exact and heuristic methods for an aeronautical assembly line time-constrained scheduling problem with multiple modes and a resource leveling objective

Tamara Borreguero Sanchidrián[a,b,*], Tom Portoleau[c], Christian Artigues[c], Alvaro García Sánchez[a,b], Miguel Ortega Mier[a,b], Pierre Lopez[c]

[a]*AIRBUS. Paseo John Lennon S/N. Getafe (28906) Spain*
[b]*Industrial Engineering and Logistics Research Group, ETSII, Universidad Politécnica de Madrid. José Gutierrez Abascal 2 (28006) Madrid*
[c]*LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France*

Abstract

Industry 4.0 means a deep transformation for all our industries. Smart factories are a key feature for this transformation. Planning and scheduling are within the core functions of those smart factories. Although scheduling problems have been studied for long, the deployment of digital scheduling solutions in industries is still an ongoing process. In this work, we discuss the detailed scheduling problem in an aeronautical assembly line. It has the structure of a challenging multimode Resource-Constrained Scheduling Problem with incompatibility constraints, a resource leveling objective and also a high number of tasks. To begin with, we present two new event-based mixed-integer linear programming formulations for this problem. Then, a constraint programming formulation is also detailed. A large-neighborhood search approach based on constraint programming and tailored to the resource leveling objective is presented. The four approaches are tested and compared using *ad hoc* data sets and industrial instances, yielding significant improvement compared to the heuristic currently used by the company. Moreover, the large-neighborhood search approach significantly improves the method proposed in the literature on a related multimode resource investment problem when short CPU times are required.

Keywords: Assembly line scheduling, multiple modes, resource leveling, mixed-integer linear programming, event-based formulation, constraint

*Corresponding author

programming, large neighborhood search

1. Introduction

The aeronautical industry has experienced an in-depth transformation in the last years. The demand for aircrafts has increased but also the complexity and customization of the products. As a result, aircraft manufacturers have had to produce more units of more complex aircrafts while trying to reduce time to market, production lead times and costs (Mas et al., 2015). In order to face these and other challenges, the aeronautical industry has moved towards the implementation of Industry 4.0 trends (Kagermann and Wahlster, 2013). As in other industries, globalization and digitization have become central features. In fact, the aeronautical industry has been ahead in the use of digital solutions. A wide range of Product Life-cycle Management tools have been deployed in the aeronautical industry and have resulted in highly digitized processes from aircraft design to aircraft maintenance (Mas et al., 2014). However, planning and scheduling processes have consistently remained almost unaffected. Most of the activities that are related to these processes continue to use manual procedures that rely on the knowledge of experts.

Nevertheless, the digitization of scheduling processes is necessary not only for Manufacturing Execution System implementation but also for Industry 4.0 deployment. Schläpfer et al. (2014) list three goals for Industry 4.0, which cannot be reached without improving planning and scheduling process:

- Higher flexibility through dynamic planning, control and execution;
- Higher productivity and resource efficiency for customized products;
- Shorter lead times through the application of intelligent analyses.

In line with these objectives, this paper proposes solution methods for the detailed scheduling of stations in an aircraft final assembly line. This problem can be classified within the Resource-Constrained Project Scheduling Problems (RCPSPs) category. In Brucker et al. (1999), a classification of such complex scheduling problems together with an overview on existing solution methods can be found. A more recent overview of RCPSP models and solution approaches can be found in Schwindt and Zimmermann (2015).

In this work, a part of which was presented in (Borreguero et al., 2015a,b), we address a multimode RCPSP with generalized temporal constraints, several labor skills and a resource leveling objective.

Our contributions can be stated as follows:

- We propose two Mixed-Integer Linear Programming event-based formulations of the problem, extending the event-based formulation initially proposed for the standard RCPSP to tackle the additional constraints coming from the considered industrial problem, which had not been studied in the literature so far.
- We also propose an exact constraint programming method to solve the problem, in order to be able to compare the performance of both types of paradigms.
- Finally, we propose a large neighborhood search heuristic with neighborhoods tailored to the resource-leveling objective, with the aim to improve the performance of exact methods and of the heuristic approach currently used by the company on large-scale industrial instances.

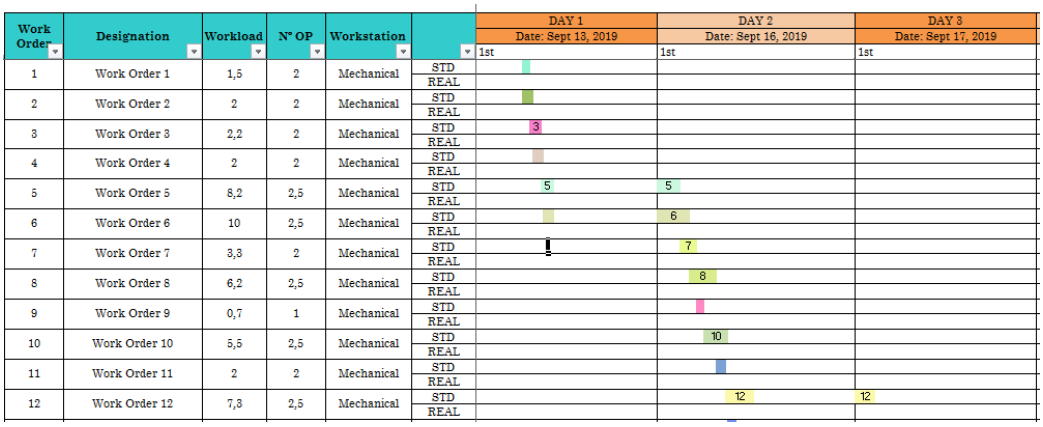
The structure of the article is as follows: Section 2 provides a more detailed problem description, Section 3 presents a literature review. Section 4 deals with the Mixed-Integer Linear Programming (MILP) formulation proposal and its experimentation results. Section 5 includes the Constraint Programming approach and its results. In Section 6, the large neighborhood search heuristic is presented. All approaches are compared, including a comparison to the existing solution methods from an aircraft manufacturer and a comparison with a method proposed in the literature for a related multi-mode resource investment problem. Finally, conclusions and further research directions are presented in Section 7.

2. Problem Description

Aeronautical assembly lines consist in a series of stations where different jobs are executed. Most of them have been transformed into moving or pulse lines where each product has to go through all the stations following a fixed path. Moreover, the line is frequently synchronized, which means that the time that each product remains on a station is always the same and equal to the rate at which the assembly line produces its output. In this context we consider the cycle time as the time needed for an airplane to visit each of the stations. In our problem the maximum cycle time is fixed.

The assignment of jobs to a station is in most of the cases related to industrial issues: assembly technologies and the need of specific jigs that cannot be easily moved. Therefore, the assignment of jobs between stations is made only once, during the line definition. Although there may be a small percentage of jobs that can be performed in more than one station, once the jobs are assigned they are rarely moved from one station to another, so we can assume with no loss of generality that the task assignment is constant per station.

Taking this into account, the scheduling decision consists in establishing the order in which the jobs will be done together with the resources allocated to each of them, given the line cycle time and a set of jobs per station. The result is usually displayed as a bar chart where each task is assigned a start/end date and a set of operators. From the operator point of view, a performance tracker view is used. The main difference is that the performance tracker is worker-oriented, so each line provides information on the task to be performed by an operator all through the cycle time. An example of each of them is presented in Figures 1 and 2.



Work Order	Designation	Workload	N° OP	Workstation		DAY 1 Date: Sept 13, 2019	DAY 2 Date: Sept 16, 2019	DAY 3 Date: Sept 17, 2019
						1st	1st	1st
1	Work Order 1	1,5	2	Mechanical	STD			
					REAL			
2	Work Order 2	2	2	Mechanical	STD			
					REAL			
3	Work Order 3	2,2	2	Mechanical	STD	3		
					REAL			
4	Work Order 4	2	2	Mechanical	STD			
					REAL			
5	Work Order 5	8,2	2,5	Mechanical	STD	5	5	
					REAL			
6	Work Order 6	10	2,5	Mechanical	STD		6	
					REAL			
7	Work Order 7	3,3	2	Mechanical	STD		7	
					REAL			
8	Work Order 8	6,2	2,5	Mechanical	STD		8	
					REAL			
9	Work Order 9	0,7	1	Mechanical	STD			
					REAL			
10	Work Order 10	5,5	2,5	Mechanical	STD		10	
					REAL			
11	Work Order 11	2	2	Mechanical	STD			
					REAL			
12	Work Order 12	7,3	2,5	Mechanical	STD		12	12
					REAL			

Figure 1: Bar Chart Example

In the recent years, automation of aeronautic stations has experienced major improvements. In spite of this, aeronautical assembly remains intensive in highly-qualified operators. Often, some of the jobs need to be done by workers with a specific certification, and not all them have the same certifications. These different skills are managed by the use of 'profiles' that gather one or more certifications. Each operator is assigned a profile, according to her/his skills. As well as this, each task may be done by operators with one or several profiles. For example, if *profile 1* includes only elementary mechanical tasks (such as drilling and rivet-

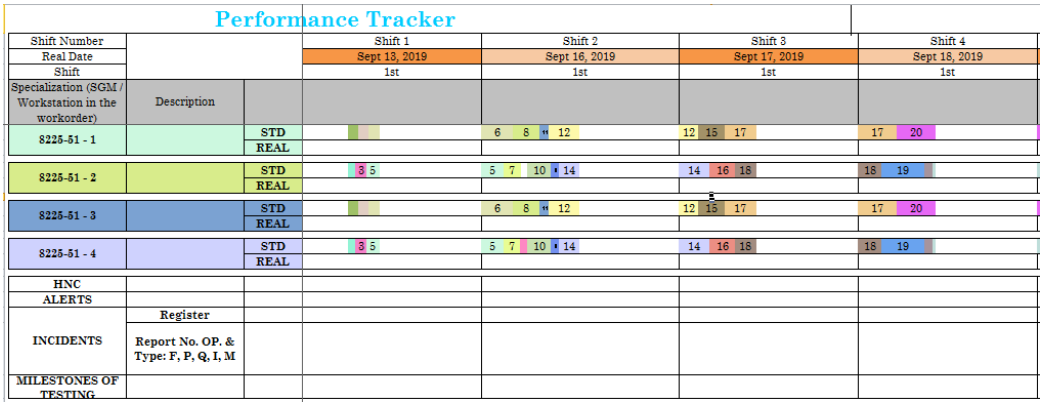


Figure 2: Performance Tracker Example

ing) and *profile 2* includes the previous ones and also pipes installation, a work involving riveting and drilling can be done by both profiles.

Operators are organized in permanent groups, with a team leader per group of 5 to 15 operators. Each group is responsible of tracking its performance and taking corrective actions. In order to enable this, each group works together and is assigned to a station for the whole cycle time. Extra resources may be available for peak demands, but they must not be included on the standard schedule. As a result, if N operators are needed only one day at one station, they will stay on the station within the cycle time, and the related cost must be allocated to that station.

Some of the assembly jobs can be performed by only one operator, whereas others must be done by two or more. On some cases, there is a range of possible operator numbers, which will lead to a set of production times for the work. In addition, having more than the minimum number of operators decreases the duration of a job, but not in a linear way, as some activities may not be performed in parallel. For example: the time for preparing the necessary material and reading the documentation will be the same no matter what is the number of operators working. As a result, the number of operators reduces the processing times in a non-linear job-specific manner. When several operators are involved, they all need to have the same profile.

Stations can be of very different sizes: from a part of a major component (a fan cowl, for example) of 1-10 meters, to a complete aircraft (hundreds of meters) in the last steps of a final assembly line. However, even the smallest stations are big enough for two or more jobs to be performed in parallel. Usually, several tasks are to be done near to each other, in a way that they cannot be executed at the

same time due to space constraints. As a result, stations are divided on smaller areas, where a limited number of operators can work at a time. In consequence, the space on each area is a scarce renewable resource for scheduling.

Each task may require a set of tools that can be standard or specific for the work. If they are specific, different jobs rarely share a tool and if they share it, the tool can be duplicated if jobs are to be performed in parallel. Standard tools are available so that all operators can use them freely. Therefore, tools need not be included in the scheduling model.

The constraints between tasks can be of different nature. The most common case is that of precedence constraints: when a task cannot be started until a previous one has been finished. For example: harnesses cannot be routed until the supports to which they are attached have been riveted to the aircraft structure, or a functional test cannot be performed until the system it tests has been completely installed.

Another kind are incompatibility or disjunctive constraints that will be called in the remaining of the paper *non-parallel* constraints: these mean that some activities can be done in any order as far as they are not being performed at the same time. This is the case of some tasks that due to health and security reasons must be done with as fewer persons as possible in the hangar, e.g. corrosion inhibition application. This also happens in tests that require a specific aircraft condition: hydraulic tests need to have the power on, but the aircraft must have its power off for fuel tests.

Finally, maximum time lags also occur. For example, bonding tests have to be performed at the end of the installations, and bonding protection must be accomplished within the same working day where the test was passed.

In accordance to the fixed cycle time, the objective function will be to minimize the resource consumption per station. As we have said, this is equivalent to minimizing the sum of the peak operators demand per profile, i.e. resource leveling.

On average, for final assembly lines, the cycle time varies from 14 to 25 working days. There are two or three operator teams per station, that is, more or less 20 operators in shifts of 8 hours. Sample profiles are: mechanical technician, electrical technician, fluid systems technician, inspector and test specialist. However, as jobs within a station are usually from similar technologies more than three profiles are rarely required per station. As for the working areas, there can be defined up to 5 different ones per station. The total workload can be from 1000 to 3000 man.hours.

Precedence constraints are the most frequent. There are normally a small per-

centage of jobs that do not have any kind of precedence relationship with the others. Jobs are usually organized on several groups (1-10) that can be done in parallel. Within these groups, most of the tasks must be done in series. Non-parallel constraints and maximal time lags occur at a much lower rate than precedence constraints but, at the same time, they cannot be omitted because they have a major impact on product quality, reliability or health and safety issues.

In summary, the short term scheduling of a final assembly line station must fulfill the following requirements:

Cycle time

- All the tasks must be completed within the stations' cycle time.

Objective function: number of operators

- Given a fixed cycle time, the aim is to find a schedule that minimizes the number of operators needed.

Station definition

- The station is divided in working areas and each operation is assigned to one of those working areas.

Task definition

- Each task can be performed in different modes. Each mode is characterized by a number and profile of operators.
- Each task can be performed by one or several operator profiles. However, all the operators selected to do the task must belong to the same profile.
- The duration of the task depends on the number of operators assigned and therefore on the selected execution mode.
- Whenever a task begins, an execution mode is selected. The same mode must be used throughout its execution.
- No preemption is allowed: once a task begins it must be continued until it is finished.

Other Constraints

- The number of operators working on an area at a specific time must be lower than the capacity of that area.
- There are precedence constraints between some operations.
- The precedence constraints between operations may also include maximal time lags.
- There are non-parallel constraints between some operations: those operations cannot be active at the same time.

Given this problem description, the next section will be dedicated to the review of similar problems in the literature, and to the most widely encountered solution approaches, including the one that is currently used by the aircraft manufacturer for the considered problem.

3. Literature Review

Concerning the relevant literature in aircraft assembly lines, assembly line balancing techniques (Boysen et al., 2009), where jobs have to be optimally assigned to stations, have been applied to aeronautical assembly lines. As a recent result, Biele and Mönch (2018) proposed mixed-integer programming techniques for assigning jobs and operators to stations in a fixed-job sequence aircraft assembly line.

Contrarily to traditional assembly line balancing, our research takes place in the production phase where jobs have already been assigned to stations, as stated in Borreguero et al. (2015c), which turns the problem into a time- and resource-constrained scheduling problem.

More precisely, we are dealing with a project scheduling problem consisting of a time window and resource assignment for a set activities of known duration and resource requests, that must be executed guaranteeing some precedence relations. Given a time limit for the project duration the objective is to find the least resource consuming schedule. Time-Constrained Scheduling Problems with resource leveling objectives are variants of Resource-Constrained Scheduling Problems. These were classified by Brucker et al. (1999) using an $\alpha|\beta|\gamma$ generalized scheme similar to the one existing for machine scheduling. In accordance with that classification, the scheduling of the tasks from an aeronautical assembly station is denoted by:

$$MPS_m, \sigma,\ \rho \mid temp \mid \sum C_o * num_o^{op}$$

- $\alpha = MPS_m, \sigma, \rho$. This stands for a multi-mode resource-constrained project where each activity can be processed in several alternative modes and there exists a set of renewable resources available for each time period during the project execution: m being the resources, σ the units of each resource available and ρ the maximum number of units of the resources demanded by an activity. For our particular problem, the activities are the work tasks assigned to each station. The renewable resources are the number of operators (each of them belonging to a profile) and the space in each of the station's working areas. As well as this, each mode for an activity defines a combination of operator profile, number of operators and duration. All the operators assigned to an activity must be from the same profile and the range of possible numbers of allocated operators per task is independent from the chosen profile.

- $\beta = temp$. There are precedence constraints (task w' cannot start until task w has been completed), non-parallel constraints (tasks w and w' cannot be in progress at the same time, but there is no precedence relation between them), and maximal time lags between tasks (task w' must start within a maximal time after w has been completed). All the temporal constraints are independent from the mode in which a task is executed.

- $\gamma = \sum C_o * num_o^{op}$. The objective function is to minimize the resource investment, given that the cycle time is fixed. In consequence, the objective function is to minimize the labor cost of the assembly. As the operators, once assigned to a station stay working on it for all the cycle time, minimizing the labor cost is in our case equivalent to minimizing the maximum number of operators needed throughout the cycle time. We will consider that the cost of an operator does not depend on its profile.

There have been a wide range of studies on both heuristic and metaheuristic methods for solving the RCPSP, as well as different MILP models (Brucker and Knust, 2011; Artigues et al., 2010; Hartmann and Briskorn, 2010; Wang et al., 2010; Nouri et al., 2013). The first MILP formulations proposed for the RCPSP were *Discrete time formulations* (Pritsker et al., 1969). Afterwards, *Continuous time formulations* were proposed by Alvarez-Valdés and Tamarit (1993) with a model based on the concept of forbidden sets and Artigues et al. (2003) with a model based on a resource flow network.

Koné et al. (2011) proposed the use of event-based formulations from a model introduced by Zapata et al. (2008) that, despite a poor LP relaxation, have the ad-

vantage to be able to tackle instances having large time horizons. They concluded that event-based formulations outperformed the time-discrete MILP models for large scheduling horizons and outperformed also the continuous time flow-based formulations for highly cumulative instances, as it is our case. However, the event-based formulations proposed by Koné et al. are suitable for the standard RCPSP, which includes some assumptions that are too restrictive for many applications (Hartmann and Briskorn, 2010). Therefore, it is of great interest to improve this kind of formulations so that they can be used on more practical RCPSP contexts. Constraint Programming (CP), on the other hand, provides a flexible and generic modeling language and efficient solutions approaches to a wide range of scheduling problems (Baptiste et al., 2001; Laborie et al., 2018). Therefore CP is a technique of choice to solve variants of RCPSP including general temporal constraints, calendar constraints (Kreter et al., 2017), multiskill operators (Polo-Mejía et al., 2020). Note that a constraint programming model was proposed by Arkhipov et al. (2018) for a closely related aircraft assembly line scheduling problem, but with the makespan minimization objective instead of the resource-leveling one.

To deal with large-scale industrial instances, while benefiting from the power of MILP or CP solvers, large neighborhood search (LNS) heuristics have been designed. This technique, based on the iterative solving of a subproblem where a part if a current solution of the global problem is fixed while another part is freed, was originally introduced in Shaw (1998) and turned out to perform well on several problems, as routing problems (Hemmelmayr et al., 2012) or scheduling problems (Palpant et al., 2004; Godard et al., 2005; Laborie and Godard, 2007; Cordeau et al., 2010; Artigues and Hébrard, 2013; Thomas and Schaus, 2018).

The LNS algorithm on variants of the RCPSP including multi-skill operators such as in Cordeau et al. (2010), generally consider time-related objective functions such as makespan, maximum lateness, sum of completion times or outsourcing costs that tend to favor compact schedules. To our knowledge, the LNS technique was never applied to multi-mode project scheduling problem with resource leveling objectives such as in our case.

The problem considered in this paper has been for long solved in the industry using expert knowledge, with the only aid of standard spreadsheet files. Recently, a scheduling tool has been implemented. It uses an activity-oriented serial scheduling generation heuristic (Borreguero et al., 2015c). Its priority rules are focused on the minimum start time per task: select the task with the smallest latest start time and smallest earliest start time. The latest and earliest start times per task are calculated using the precedence diagram and the resource availability. This heuristic is good at providing feasible solutions fast enough, given a

station's cycle time and a resource availability. However, there is no proof of optimality in terms of resources. In consequence, the scheduler needs to follow an iterative approach, which requires a long time and, in most cases, ends up with non-optimal solutions. Therefore, the aim of this paper is to investigate the use of mixed-integer linear programming and constraint programming models, which may prove optimality of the solution reached on some instances. Also, an efficient large neighborhood search techniques able to tackle the resource leveling objective will be presented.

4. MILP Formulation

We have developed two new event-based formulations for the aeronautical station scheduling problem. They are inspired by the Start/End and On/Off Event-Based Formulations presented by Koné et al. (2011).

Extension of previously existing event-based models to our problem is not immediate. The main binary decision variables have been modified with two new sub-index in order to deal with the multiple modes per task. Also, the original formulations included only standard precedence constraints. New constraints and variables have been added in order to take into account the maximal time lag and non-parallel constraints. Finally, the objective function has been modified to tackle with the resource leveling objective. In the original event-based formulations, the number of events were, at most, the number of scheduled tasks plus one. In our case, we need to add one more event for each pair of tasks with maximal time lag precedence.

Both formulations include four different sets. Set W includes the job tasks, set P the operator profiles, and set A the station areas. Finally, due to the formulation chosen, we will need a set $E = \{0, 1, ..., n\}$ of events. The other needed sets and parameters are explained in Table 1.

4.1. Start/End Event-based Multimode Formulation: SEE-M

Two sets of variables, x_{weop} and y_{weop}, are used to define the start and end events of each task, as $x_{weop} = 1$ ($y_{weop} = 1$) if task $w \in W$ starts (ends) at event $e \in E$ using o operators of profile $p \in P_w$.

As already mentioned, significant changes have to be brought to the original Start/End event-based model to take into account the constraints coming from the industrial problem: for example, to guarantee the same assignation of operator profiles and number throughout the task processing. Also, for tasks involved on some non-parallel constraint we need to define a continuous variable, S_w which

Table 1: Event-Based Formulation Parameters

P_w	Set of operator profiles that may perform task w, $\forall w \in W$
MX_w	Maximum number of operators that can work on task w, $\forall w \in W$
MN_w	Minimum number of operators that can work on task w, $\forall w \in W$
M_w	Set of possible operator numbers that can work on task $M_w = \{MN_w, \ldots, MX_w\}$
D_w	Total duration (in working hours) of task $w \in W$, if assigned only to one operator, $\forall w \in W$
Γ_{opw}	Reduction coefficient to obtain task w duration when it is done by o of operators with operator profile p, $\forall w \in W$, $o \in N_w$
A_{aw}	1 if task w is done on area a, 0 otherwise, $\forall a \in A$, $w \in W$
PR	Set of precedence relationships between tasks: $(w, w') \in PR$ means that $w \in W$ must precede $w' \in W$
NP	Set of non-parallel constraint between tasks: $(w, w') \in NP$ means that $w \in W$ cannot be scheduled in parallel with $w' \in W$
MT	Set of maximal time lag constraints: $(w, w') \in MT$ means that the start time of w' must not exceed the end time of w plus a fixed time lag Δ
WM	Set of tasks $w \in W$ involved in a maximal time lag constraint: $\exists w' \in W$, $\{(w', w), (w', w)\} \cap MT \neq \emptyset$
C_a	Area capacity (in terms of number of operators), $\forall a \in A$
CT	Station cycle time

Table 2: SEE-M Variables

x_{weop}	1 if task w starts at event e with o operators of profile p and 0 otherwise, $\forall w \in W, e \in E \setminus \{n\}, o \in M_w, p \in P_w$
y_{weop}	1 if task w ends at event e with o operators of profile p and 0 otherwise, $\forall w \in W, e \in E \setminus \{0\}, o \in M_w, p \in P_w$
r^*_{pe}	Number of operators of profile p required by the tasks in progress immediately after event e, $\forall p \in P, e \in E$
s^*_{ae}	Number of operators on area a required by the tasks in progress immediately after event e, $\forall a \in A, e \in E$
$\alpha_{ww'}$	1 if w ends before w' starts and 0 vice-versa. Defined $\forall (w, w') \in NP$
t_e	Time of event $e \in E$
n_p	Total number of operators of profile p needed, $p \in P$
S_w	Start time of task w. Used for maximal time lag constraints, and therefore defined only $\forall w \in WM$

represents the starting time of task w. Another new set of variables, n_p, represents the total number of operators of profile $p \in P$ needed. All the model variables are listed in Table 2, where $first(E)$ and $last(E)$ denote the first and last event of event set E, respectively.

The complete SEE-M formulation is as follows:

$$Minimize \sum_{p \in P} n_p \tag{1}$$

Subject to:

$$t_0 = 0 \tag{2}$$

$$t_e \leq CT \quad \forall e \in E \tag{3}$$

$$t_{e+1} - t_e \geq 0 \quad \forall e \in E \setminus \{n\} \tag{4}$$

$$\sum_{e \in E, o \in M_w, p \in P_w} e y_{weop} - \sum_{e \in E, o \in M_w, p \in P_w} e x_{weop} \geq 1 \quad \forall w \in W \tag{5}$$

$$\sum_{e \in E, o \in M_w, p \in P_w} x_{weop} = 1 \quad \forall w \in W \tag{6}$$

$$\sum_{e \in E, o \in M_w, p \in P_w} y_{weop} = 1 \quad \forall w \in W \tag{7}$$

$$t_f - t_e - \sum_{o \in M_w, p \in P_w} D_w \Gamma_{opw} (x_{weop} + 1 - y_{wfop}) \geq 0 \quad \forall (f, e) \in E,$$

$$f > e, w \in W \tag{8}$$

$$\sum_{e\in E, p\in P_w} x_{weop} = \sum_{e\in E, p\in P_w} y_{weop} \quad \forall w \in W, o \in M_w \tag{9}$$

$$\sum_{e\in E, o\in M_w} x_{weop} = \sum_{e\in E, o\in M_w} y_{weop} \quad \forall w \in W, p \in P_w \tag{10}$$

$$\sum_{\substack{e''=0 \\ o\in M_{w'}, p\in P_{w'}}}^{e-1} x_{w'eop} + \sum_{\substack{e'=e \\ o\in M_w, p\in P_w}}^{n} y_{we'op} \leq 1 \quad \forall e \in E, (w, w') \in PR \tag{11}$$

$$S_w \geq t_e - CT(1 - \sum_{o\in M_w, p\in P_w} x_{weop}) \quad \forall w \in WM \tag{12}$$

$$S_w \leq t_e + CT(1 - \sum_{o\in M_w, p\in P_w} x_{weop}) \quad \forall w \in WM \tag{13}$$

$$S_{w'} - S_w - \sum_{e\in E, o\in M_w, p\in P_w} \Gamma_{opw} D_w x_{weop} \leq \Delta \quad \forall (w, w') \in MT \tag{14}$$

$$\sum_{e\in E, o\in M_w, p\in P_w} ey_{weop} - \sum_{e\in E, o\in M_{w'}, p\in Pw'} ex_{w'eop} \leq n(1 - \alpha_{ww'}) \quad \forall (w, w') \in NP \tag{15}$$

$$\sum_{e\in E, o\in M_{w'}, p\in P_{w'}} ey_{w'eop} - \sum_{e\in E, o\in M_w, p\in P_w} ex_{weop} \leq n(\alpha_{ww'}) \quad \forall (w, w') \in NP \tag{16}$$

$$r^*_{p0} - \sum_{w\in W, o\in M_w} ox_{w0op} = 0 \quad \forall p \in P \tag{17}$$

$$r^*_{pe} - r^*_{pe-1} + \sum_{w\in W, p\in P_w} (\sum_{o\in M_w} oy_{weop} - \sum_{o\in M_w} ox_{weop}) = 0 \quad \forall o \in O, e \in E \setminus \{0\} \tag{18}$$

$$r^*_{pe} \leq n_p \quad \forall p \in P, e \in E \tag{19}$$

$$s^*_{a0} - \sum_{w\in W, o\in M_w, p\in P_w} ox_{w0op} A_{aw} = 0 \quad \forall a \in A \tag{20}$$

$$s^*_{ae} - s^*_{ae-1} + \sum_w (\sum_{o\in M_w, p\in P_w} oy_{weop} A_{aw} - \sum_{o\in M_w, p\in P_w} ox_{weop} A_{aw}) = 0 \quad \forall a \in A, e \in E \setminus \{0\} \tag{21}$$

$$s^*_{ae} \leq C_a \quad \forall a \in A, e \in E \tag{22}$$

$$x_{weop} \in \{0, 1\} \quad \forall e \in E \setminus \{0\},\ w \in W$$

$$p \in P_w, o \in M_w \quad (23)$$

$$y_{weop} \in \{0,1\} \quad \forall e \in E \setminus \{n\},\ w \in W$$

$$p \in P_w, o \in M_w \quad (24)$$

$$\alpha_{w,w'} \in \{0,1\} \quad \forall (w,w') \in NP \quad (25)$$

$$r^*_{pe} \geq 0 \quad \forall e \in E,\ p \in P \quad (26)$$

$$0 \leq s^*_{ae} \leq C_a \quad \forall e \in E, a \in A \quad (27)$$

$$S_w \geq 0 \quad \forall w \in WM \quad (28)$$

$$t_e \geq 0 \quad \forall e \in E \quad (29)$$

The objective function (1) is to minimize the total project cost. This cost depends only on the labor costs, which are proportional to the maximum number of operators needed during the planning horizon, as resources allocated to a work station will remain in it throughout the complete cycle time. Constraint (2) forces the first event to begin at $t = 0$ and Constraint (3) ensures that there is no delay in the station completion. The order of the events on time is imposed by Constraint (4). Constraint (5) states that the start event of a task must precede its end event. Constraints (6) and (7) limit to one the start and end event per work task. Constraint (8) fixes the minimum time difference between the start and the end events to the duration of the task.

A single mode for performing the task is imposed by Constraints (9) and (10). Note that the possible modes are limited by the definition of x_{weop} and y_{weop} as they are only defined for $o \in M_w$ and $p \in P_w$, see (23).

As for the relations between tasks, Constraint (11) is the multimode expression for the precedence constraints. It states that if w must precede w' then, if w finishes at event e, w' cannot start until event $e+1$. Maximal time lags are enforced by Constraints (12) to (14). Constraints (12) and (13) define the start time of a task. These constraints will only be calculated for the task involved on maximal time lag constraints (set WM) and it classically involves the cycle time CT as a big-M coefficient to withdraw the constraint in case there is no precedence between the tasks. Constraints (14) limit the time between the end of a task and the start of its successor. This time lag is defined as a constant parameter, in our case Δ (although it could be easily replaced by a task dependent parameter $\Delta_{ww'}$). Constraints (15) and (16) define the non-parallel constraints. In these constraints, n (the number of events) is used as a big-M parameter.

As for the resource consumption, Constraint (17) computes the number of operators needed per profile immediately after the first event. Constraint (18) computes the number p of operators per profile, for all the other events. Constraint (19)

computes the maximum need of operators per profile n_p over all events. Finally, Constraints (20) and (21) ensure that the maximum occupation in the station areas is never exceeded.

4.2. On/Off Event-based Multimode Formulation: OOE-M

The other event-based formulation we have implemented deals with a single set of variables z_{weop}, whose value is 1 if task w is active from event e to event $e+1$ with o operators of profile p and 0 otherwise. Another new set of binary variables, β_{wop} will be used to choose the mode in which a task is performed. As resources are modeled in a simpler way, r^*_{oe} and s^*_{ae} are no longer needed. There is no need either for variables $\alpha_{ww'}$ for the non-parallel constraints. Variables t_e, n_p and S_w are common to the SEE-M model.

The OOE-M formulation is:

$$Minimize \sum_{p\in P} n_p \tag{30}$$

Subject to:

$$t_0 = 0 \tag{31}$$

$$\sum_{w\in W, o\in M_w, p\in P_w} z_{w0op} \geq 1 \tag{32}$$

$$t_{e+1} - t_e \geq 0 \quad \forall e \in E \setminus \{n\} \tag{33}$$

$$\sum_{e\in E, o\in M_w, p\in P_w} z_{weop} \geq 1 \quad \forall w \in W \tag{34}$$

$$\sum_{p\in P_w, o\in M_w} \beta_{wop} = 1 \quad \forall w \in W \tag{35}$$

$$z_{weop} \leq \beta_{wop} \quad \forall w \in W, p \in P_w, e \in E, o \in M_w \tag{36}$$

$$\sum_{\substack{e'=0 \\ o\in M_w, p\in P_w}}^{e-1} z_{we'op} - e(1 - \sum_{o\in M_w, p\in P_w} (z_{weop} - z_{we-1op})) \leq 0 \quad \forall w \in W, e \in E \tag{37}$$

$$\sum_{\substack{e'=e \\ o\in M_w, p\in P_w}}^{n-1} z_{we'op} - (n-e)(1 + (\sum_{o\in M_w, p\in P_w} (z_{weop} - z_{we-1op}))) \leq 0 \quad \forall w \in W, e \in E \tag{38}$$

$$t_f - t_e - \sum_{o\in M_w, p\in P_w} D_w \Gamma_{opw} (z_{weop} - z_{we-1op} - (z_{wfop} - z_{wf-1op}))$$

$$\geq -\sum_{e' \in E, o \in M_w, p \in P_w} (D_w \Gamma_{opw} z_{we'op}) \quad \forall (f,e) \in E, \; f > e, w \in W \quad (39)$$

$$CT - t_e - \sum_{o \in M_w, p \in P_w} D_w \Gamma_{ow} (z_{weop} - z_{we-1op}) \geq 0 \quad \forall w \in W, \; e \in E \quad (40)$$

$$\sum_{o \in M_w, p \in P_w} z_{weop} + \sum_{\substack{e'=0 \\ o \in M_{w'}, p \in P_{w'}}}^{e} z_{w'e'op} - (e-1)(1 - \sum_{op} z_{weop}) \leq 1 \quad \forall e \in E, \; (w,w') \in PR \quad (41)$$

$$S_w \geq t_e - CT(1 + z_{we-1op} - z_{weop}) \quad \forall e \in E, \; w \in WM \quad (42)$$

$$S_w \leq t_e + M(1 + z_{we-1op} - z_{weop}) \quad \forall e \in E, \; w \in WM \quad (43)$$

$$S_{w'} - (S_w + \sum_{o \in M_w, p \in P_w} \beta_{wop} D_w \Gamma_{opw}) \leq \Delta \quad \forall (w,w') \in MT \quad (44)$$

$$\sum_{o \in M_w, p \in P_w} z_{weop} + \sum_{o \in M_{w'}, p \in P_{w'}} z_{w'eop} \leq 1 + n(1 - NP_{ww'}) \quad \forall (w,w') \in NP \quad (45)$$

$$\sum_{w \in W, o \in M_w} o z_{weop} \leq n_p \quad \forall p \in P, e \in E \quad (46)$$

$$\sum_{w \in W, o \in M_w, p \in P_w} o z_{weop} A_{aw} \leq C_a \quad \forall a \in A, e \in E \quad (47)$$

$$z_{ewop} \in \{0,1\} \quad \forall e \in E, w \in W, \; p \in P_w, o \in M_w \quad (48)$$

$$z_{w-1op} = 0 \quad \forall w \in W, p \in P_w, \; o \in M_w \quad (49)$$

$$\beta_{wop} \in \{0,1\} \quad \forall w \in W, p \in P_w, \; o \in M_w \quad (50)$$

$$t_e \geq 0 \quad \forall e \in E \quad (51)$$

$$S_w \geq 0 \quad \forall w \quad (52)$$

The objective function is to minimize the total project cost (30). The first event on the project starts at $t = 0$ per Constraint (31) and at least one task must be active after this event as per Constraint (32). Constraint (33) refers to the order of the events, allowing two of them to occur at the same time.

Each task must be active at least after one of the events (34) in order to ensure the scheduling of all the tasks. Variables β_{wop} select the mode in which each task will be performed. One and only one of the variables β_{wop} can be set to 1 per task, Constraint (35) and the tasks can only be performed on the selected mode, as per (36).

Constraints (37) to (40) are based on the three values than can take the difference $z_{weop} - z_{we+1on}$:

- $z_{weop} - z_{we+1op}$= -1. When $e+1$ is the first event after which w is active, so $z_{weop} = 0$ and $z_{we+1op} = 1$.
- $z_{weop} - z_{we+1op}$=1. When e is the last event after which w is active, so $z_{weop} = 1$ and $z_{we+1op} = 0$.
- $z_{weop} - z_{we+1op}$=0. Otherwise

Constraints (37) and (38) refer to the continuous processing of each task: by (37) if task w begins after event e, then it cannot be processed before $e-1$. Similarly, by (38) if task w ends at event e then w is no longer active $\forall e' \geq e+1$. The time of a task is measured by the difference between the start event (the first event e after which w is active) and the end event (the first the last event after which w is active).

The time difference between w's start event and its end event must be at least the work task's processing time (39) and none of the tasks can end after the station cycle time, see (40).

Regular precedence constraints are (41), as if w must precede w', then it must start at an event after which w is no more active. Maximal time lags are expressed by Constraints (42) to (44). Constraints (42) and (43) define the start time of a task. These constraints will only be calculated for the tasks involved on maximal time lag constraints. Together with them, Constraints (44) limits the time between the end of a task and its successor's start time. Non-parallel constraints are (45), as two non-parallel tasks cannot be active at the same time.

Resource constraints are simpler than for the SEE-M formulation. In this case, only one set of constraints is defined per scarce resource: (46) for the quantity of operators per type and (47) for the amount of operators per area, and no specific variables are defined for these constraints.

Note that there are less binary variables than in the SEE-M formulation. As for the number of constraints, for precedence and maximal time lag constraints, both models are similar. The SEE-M has twice as constraints as the OOE-M for

Table 3: Instance Characteristics

Set	$\|W\|$	$\|PR\|$	$\|MT\|$	$\|NP\|$	$\|P\|$	$\|A\|$	$\sum_{w \in W} M_w$
Set1-8	8	6	1	1	2	2	12
Set2-8	8	8	1	1	2	2	16
Set3-8	8	7	1	1	2	2	17
Set4-8	8	7	1	1	2	2	16
Set3-9	9	8	1	1	2	2	18
Set3-10	10	9	1	1	2	2	18
Set3-11	11	10	1	1	2	2	18
Set4-9	9	8	1	1	2	2	16
Set4-10	10	9	1	1	2	2	16
Set4-11	11	10	1	1	2	2	16

defining the non-parallel constraints. On all, it is not possible to establish a general dominance rule.

4.3. MILP Experimentation Results

The MILP models were validated and compared on randomly generated instances. The computational results were obtained using CPLEX12.4 solver. The tests were carried out on an Intel Core i7 2630QM processor with 2 GHz and 4 GB RAM, running Windows 7. Four sets of small 8-task instances were used. Moreover, Sets 3 and 4 were extended in order to create instances of up to 11 tasks. Their characteristics are listed in Table 3, which gives, for each instance family, the number of tasks, the number of precedence constraints, the number of maximal time lag constraints, the number of non-parallel constraints, the total number of operator profiles, the number of areas, and the total number of different modes.

Overall, 75 different combinations of data sets, number of tasks, cycle time and number of events were tested for each formulation.

All instances were solved up to optimality for both formulations. For SEE-M formulation, the instances took times from seconds to fifteen minutes, and from 0.1 second to eight minutes for the OOE-M formulation. The harder instance to solve was Set3-11, that took up to 4133 seconds for a $CT = 17$ days and 11 events on the SEE-M formulation and 452 on the OOE-M formulation.

Both formulations have similar behavior as far as the impact of variations on the number of events, number of tasks and cycle time on the solution time. As for

the evolution of the solution time throughout different cycle times, on average the solution time also grew as the objective cycle time got closer to the critical path length. Table 4 shows an example of this evolution for each of the formulations.

Table 4: Sample Solving Time for Different Cycle Times

	Instance	CT 31.5	CT 33	CT 34.75	CT 41
SEE-M	Set2-8	10.2s	6.65s	1.79s	0.83s
OOE-M	Set2-8	4.66s	0.83s	0.47s	0.2s

Finally, focusing on the influence of the number of tasks, most of the instances required more solution time with the same number of events when new tasks were added. It must be noted that some instances were solved faster with more tasks. This shows that in some cases the combinatorial structure of the problem is more important than the number of tasks itself. We also stated that the hardening of the instances as we add new tasks is wider on the cases where we are using more events.

However, the major impact on the models' performance is related to the number of events used to solve an instance. For each set and cycle time, different number of events were tested. Starting from the theoretical minimum number of events, they were reduced until solutions where no longer optimal. The solution time increases exponentially with the number of events, even when solving the same set of instances.

Figure 3 shows the evolution of the SEE-M solution times for some of the instances whenever the number of events changed. On this figure, the series data include information on the data set, the number of tasks and the input cycle time: Set1-8-11.5 stands for the solution of data set 1, with 8 tasks and $CT = 11.5$ days. For example, for the SEE-M formulation the first set (Set1-8), when solved for $CT = 11.5$ days took from 2 to 281 seconds, depending on the number of events. The evolution of the solution time against the number of events follows a similar pattern for the OEE-M model.

Taking into account that in most of the instances the precedence graph had at least two parallel paths, the possibility of event reduction could have been foreseen. Another important factor is the relationship between the shortest possible cycle time and the objective cycle time, as a tighter cycle time would lead to more tasks to be performed in parallel and therefore less events to be needed.

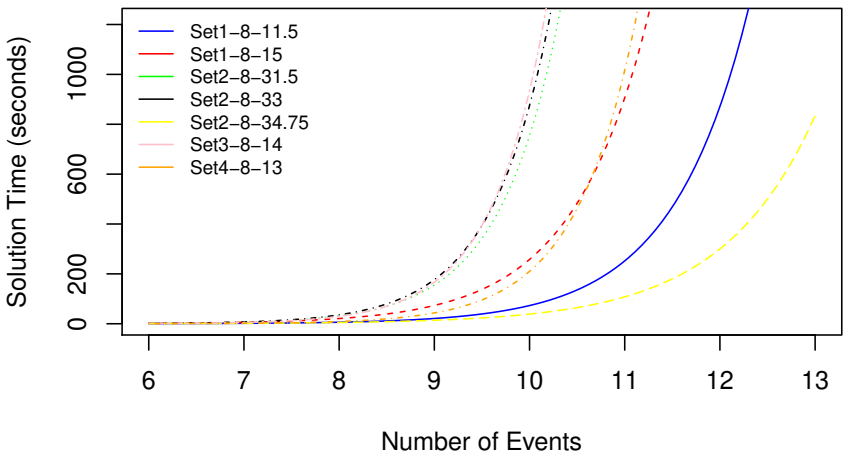


Figure 3: SEE-M Solution Time *vs.* Number of Events

As for the comparison between the two formulations, the results in terms of solution time, number of nodes and first lower bound have been better for all the instances with the OOE-M formulation than with the SEE-M formulation. Figure 4 shows the histogram for the division of the time spent for a solution with the OOE-M formulation between the time spent by the SEE-M formulation. The only two cases where the solving time is longer for the OOE-M formulation are instances with solution times within the range of 0.5 seconds, where the absolute difference is not relevant.

In fact, the difference between both formulations grows with the number of events. Therefore, as the complexity of the instances grows the use of the OOE-M formulation becomes more suitable.

These comparative results are coherent with the results of Koné et al. (2011) for the single mode problem with only precedence constraints, who concluded that the OOE outperformed the SEE formulation for all the instance sets.

5. Constraint Programming

In the previous section, we showed how to extend the event-based formulations proposed in Koné et al. (2011) for the RCPSP to the considered multi-mode

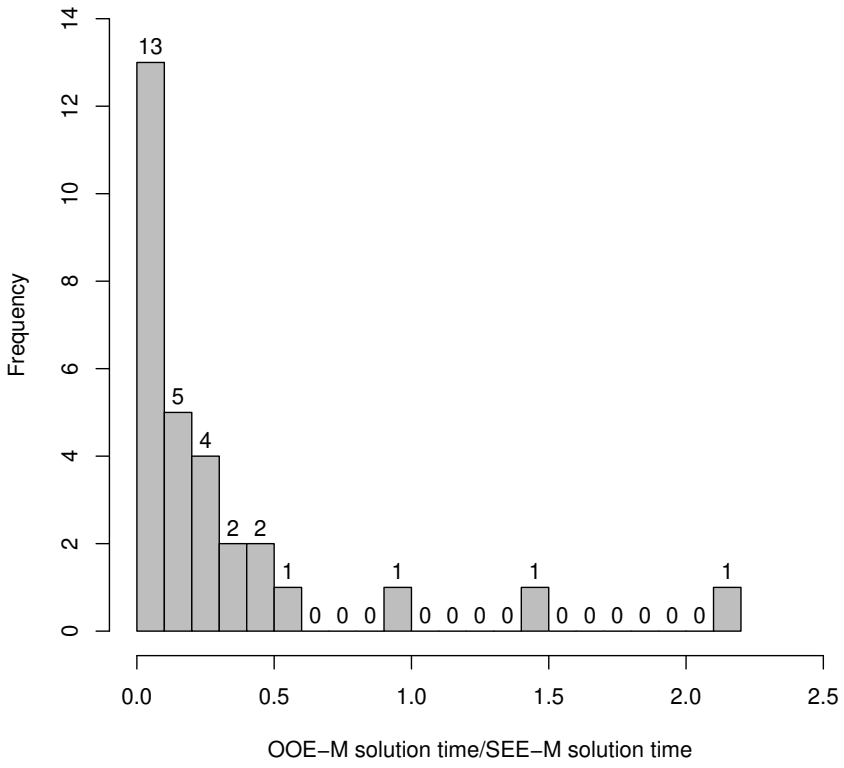


Figure 4: OOE-M Solution Time / SEE-M Solution Time

industrial problem with maximal time lags, non-parallel constraints and a resource leveling objective. However, the obtained models could only be used on small instances. Constraint Programming (CP) has been proven to be an efficient method on several combinatorial optimization problems, especially scheduling problems (Baptiste et al., 2001). Therefore, we propose a CP model based on standard scheduling constraints.

5.1. Constraint Programming Model

For modeling and solving, we use the CP Optimizer constraint-based scheduling library (Laborie et al., 2018). Below we refer to the basic modeling elements we use. We refer to Laborie et al. (2018) for a more detailed definition of these

elements.

- A **task** $w \in W$ is modeled as an `interval` decision variables T_w with a release date 0 and a due date CT. In the CP Optimizer modeling language this is written, for all $w \in W$:

$$\texttt{dvar interval } T_w \texttt{ in } 0..CT$$

- **Modes**: A mode alternative for operator profile $p \in P_w$ and a number of operators $o \in M_w$ is an `optional` interval variable T_{wop}, which is written, for all $w \in W$, $p \in P_w$ and $o \in M_w$:

$$\texttt{dvar interval optional } T_{wop} \texttt{ in } 0..CT \texttt{ size } \Gamma_{opw} D_w$$

  Each task $w \in W$ is an `alternative` between the optional mode tasks:

$$\texttt{alternative}(T_w, (T_{wop})_{p \in P_w, o \in M_w})$$

- **Precedences**: The standard precedence constraints between a task w and its successor w', for each $(w, w') \in PR$ is written:

$$\texttt{endBeforeStart}(T_w, T_{w'})$$

  Similarly a maximal time lag constraint $(w, w') \in MT$ is defined by:

$$\texttt{startBeforeEnd}(T_{w'}, T_w, -\Delta)$$

- **Resource constraints**: Task consumption on a resource is modeled as a `pulse` function, equal to 0 outside the execution interval of a task and equal to the resource usage (here the number of operators used by the task in its selected mode) when the task is in process. The total consumption on a resource is a sum of `pulse` functions:

  – for each area $a \in A$, $\sum_{\substack{w \in W | A_{aw}=1, \\ p \in P_w, o \in M_w}} \texttt{pulse}(T_{wop}, o) \leq C_a$

  – for each profile of operators $p \in P$, $\sum_{w \in W, o \in M_w} \texttt{pulse}(T_{wop}, o) \leq n_p$

Table 5: Real Instance Characteristics

Set	$\|W\|$	$\|PR\|$	$\|MT\|$	$\|NP\|$	$\|P\|$	$\|A\|$	$\sum_{w\in W} M_w$
Set1-70	70	64	0	11	4	2	84
Set1-100	100	90	3	14	4	2	134

- **Incompatibility (non-parallel) Constraints**: For each pair (w, w') of incompatible tasks a `noOverlap` constraint is used:

$$\texttt{noOverlap}(T_w, T_{w'})$$

- **Objective**: $\min \sum_{p\in P} n_p$

5.2. Experimental Results and Comparison with the MILP Approaches

Experimentation on Constraint Programming was performed on a PC DELL Inspiron 1525 Intel(R) Core(TM)2 Duo CPU, T5550 @ 1.83 GHz and 3.0 Go RAM. The instances used have been on the one side, a selected number of instances common to the MILP experimentation and, on the other, two real instances from the Aircraft Final Assembly Line. The real instances characteristics are summarised in Table 5. Due to the use of CP Optimizer 12.6.0, the time horizon has been scaled in each instance in order to obtain integer durations. This is not an issue for real application, as integer durations (expressed in minutes) can be used with no loss of generality. The default search mode of CP Optimizer was used.

All the instances have been solved up to optimality. Small instances took less than a second, whereas bigger instances were solved in at most 20 seconds while the MILP model could not obtain feasible solutions for the larger instances. Solution times for the MILP and CP models are listed in Table 6. The results clearly establish the superiority of the CP Optimizer solver compared to the MILP models on the considered scheduling problem.

6. Large Neighborhood Search Method

6.1. The Large Neighborhood Search Technique

In the previous sections, we introduced MILP and CP formulations with the aim of solving exactly our problem. However, since this problem is NP-hard, exact methods are not necessarily appropriate. The larger instance solved by the CP model in 20 seconds (see Table 6) has 100 tasks while the real aircraft assembly scheduling problem has more than 700 tasks. In this section, we propose

Table 6: MILP - CP Optimizer Comparison

Inst	CT	SEE-M (STD) t(s)	OEE-M (STD) t(s)	OEE-M (MIN) t(s)	CP optimizer t(s)
Set1-8	11.5	21.09	1.93	0.67	0.01
Set2-8	34.75	19.66	0.58	0.47	0.02
Set3-8	14	121.98	8.24	0.22	0.13
Set4-8	13	36.58	3.82	0.11	0.02
Set3-11	17	4133	452.53	1.45	0.29
Set4-11	13	473.54	76.32	13.07	0.05
Set1-70	1200	—	—	—	2.3
Set1-100	1700	—	—	—	20.7

a heuristic method based on Large neighborhood Search (LNS) technique that we evaluate against the CP Optimizer solver and the heuristic used in practice on industrial instances by the aircraft manufacturer. The main principle of LNS techniques, inspired by Palpant et al. (2004), is the following:

(0) Compute an initial solution $\mathscr{S}$ of the problem $\mathscr{P}$.

(1) Fix a part of solution $\mathscr{S}$ such that the unfixed part is critical w.r.t. the objective function.

(2) Compute a new solution $\mathscr{S}'$ for the problem $\mathscr{P}'$, where $\mathscr{P}'$ is a problem issued from $\mathscr{P}$ with the constraints induced by step (1).

(3) If $\mathscr{S}'$ is better than $\mathscr{S}$ then $\mathscr{S} \leftarrow \mathscr{S}'$.

(4) If the stop condition is not met go to step (1).

(5) Return $\mathscr{S}$.

The LNS generic principles do not specify any type of diversification, but rather rely on the idea that if the neighborhood of a solution is large enough, the quality of local optima in the neighborhood tends to be better.

Clearly, one the main stake of LNS algorithms is deciding which part of the initial solution to fix at step (2) so that its neighborhood contains better solutions; in other words, how to evaluate the criticality of a solution part?

6.2. *Large Neighborhood Search for the Aircraft Assembly Line Scheduling Problem*

As mentioned in literature review, LNS is generally applied to scheduling problems where the objective is to minimize a time-related criterion or an outsourcing cost. A typical large neighborhood of a solution in this context consists in selecting a time interval and fixing all activities scheduled outside the interval and compacting as much as possible the activities scheduled inside the interval. This is the case for the first LNS method proposed for the RCPSP (Palpant et al., 2004). Notably, the default search of the IBM CP Optimizer we used in the previous section also implements an LNS method based on this principle (Laborie and Godard, 2007). This is not what we should do for the considered problem, as compacting a schedule as much as possible would inevitably increase the resource usage.

In the problem considered here, we aim at minimizing the maximal use of a given resource. We aim at identifying the set of tasks that are involved in the maximal resource peaks. Consider a solution $\mathscr{S}$ where each task $w \in W$ has start time $\bar{S}_w \in [0, CT]$, a number of assigned operators $\bar{o}_w \in M_w$ for operator profile $\bar{p}_w \in P_w$ and a maximal number of operators $\bar{n}_p$ for each operator profile $p \in P$. The set of peak tasks is the set of all critical sets as defined below:

Definition 1. *A critical set $\tilde{W}$ is a set of overlapping tasks that reaches the maximal number of operators for at least one profile $p \in P$. More formally: $\exists t \in [0, CT]$, $\exists p \in P$, $\forall w \in \tilde{W}$, $\bar{p}_w = p$, $\bar{S}_w \leq t < \bar{S}_w + \Gamma_{\bar{o}_w \bar{p}_w w} D_w$ and $\sum_{w \in \mathscr{C}} \bar{o}_w = \bar{n}_p$.*

In fact, the resource usage only changes at the beginning or the end of a task. Let $\mathscr{T}$ denote the set of different start and end time of the tasks. The set of all critical sets can be enumerated by a sweep algorithm that tests the condition of definition 1 for each set built by the task that overlaps each time point in $\mathscr{T}$. Algorithm 1 describes the sweep algorithm that computes the set of all peak tasks $\mathscr{C}$ in $\mathscr{O}|W|^2|P|$ time.

In order to generate a high quality neighborhood, we let free all the tasks that contribute to the maximal use of the objective resource (the ones belonging to the peak set computed by the `sweep` algorithm) and the tasks that must precede them by a precedence constraint, and we fix the others.

We then solve this new problem given the bound provided by the value of the initial solution and the constraints induced by the fixed tasks, within a limited time. If a solution has been found, it replaces the initial solution as the best solution and we start over. However, if no solution was found, we solve a new problem, fixing fewer tasks and setting a greater solving time, using the self-adaptive

Algorithm 1 The `sweep` algorithm for peak task computation

Require: A problem $\mathscr{P}$ and a solution $\mathscr{S} = \{(\bar{S}_w, \bar{p}_w, \bar{o}_w)_{w \in W}, (\bar{n}_p)_{p \in P}\}$
  $\mathscr{C} \leftarrow \emptyset$
  $\mathscr{T} \leftarrow \{\bar{S}_w | w \in W\} \cup \{\bar{S}_w + \Gamma_{\bar{o}_w \bar{p}_w w} D_w | w \in W\}$
  for $p \in P$ **do**
    for $t \in \mathscr{T}$ **do**
      $\tilde{W} \leftarrow \emptyset$; $cons \leftarrow 0$
      for $w \in W$ **do**
        if $\bar{p}_w = p$ **and** $\bar{S}_w \leq t < \bar{S}_w + \Gamma_{\bar{o}_w \bar{p}_w w} D_w$ **then**
          $\tilde{W} \leftarrow \tilde{W} \cup \{w\}$; $cons \leftarrow cons + \bar{o}_w$
        end if
      end for
      if $cons = \bar{n}_p$ **then**
        $\mathscr{C} \leftarrow \mathscr{C} \cup \tilde{W}$
      end if
    end for
  end for
  return $\mathscr{C}$

principle originally proposed by Palpant et al. (2004). To be more specific, each time the solver is unable to find a solution, we fix 10% less activities and add 10 seconds to the maximum solving time. These values were determined empirically using the instances from the benchmark considered in the previous sections.

In our implementation, we use CP Optimizer as a black box to solve different generated subproblems using the constraint programming model described in Section 5. Algorithm 2 provides the pseudo-code of our implementation of the LNS method for the aircraft assembly line scheduling problem.

Note that `presenceOf`(T_{wop}) in the CP Optimizer language is a constraint that enforces the presence of the optimal task T_{wop}, while `startAt`(T_w, t) is a constraint that fixes the start time of task T_w to value t. These two constraints are used to fix the modes and the start times of the tasks in $W \setminus W'$ while the tasks in W' are freed and form the LNS subproblem. Note that τ' is a value much lower than τ giving the amount of time devoted to the CP solver to get an initial solution.

6.3. Experimental Comparison with CP Optimizer and the Heuristic Used in Practice

In order to assess the LNS algorithm efficiency as an heuristic we use a set of instances different from those dealt in the previous sections. These instances

Algorithm 2 LNS for the aircraft assembly line scheduling problem

Require: An aircraft assembly line scheduling problem $\mathscr{P}$ in the form of a constraint programming model (Section 5.1) and a time limit τ

Initialize solution $\mathscr{S}^* = \{(S_w^*, p_w^*, o_w^*)_{w \in W}, (n_p^*)_{p \in P}\}$ by solving $\mathscr{P}$ with CP Optimizer under time limit τ'

$\pi_{ratio} \leftarrow 100$

$\tau_{base} \leftarrow 10$

$\tau_{inc} \leftarrow 0$

while elapsed time $< \tau$ **do**

  $\tilde{W} \leftarrow \texttt{sweep}(\mathscr{P}, \mathscr{S}_{best})$ (get the peak tasks)

  $W^* \leftarrow \tilde{W} \cup \{W' \in W | (w', w) \in PR\}$ (add the tasks that precede them)

  $W' \leftarrow$ a subset of W^* where we randomly select $\pi_{ratio}\%$ tasks

  $\mathscr{P}' \leftarrow \mathscr{P}$

  for $w \in W \setminus W'$ **do**

    $\mathscr{P}' \leftarrow \mathscr{P}' \cup \texttt{presenceOf}(T_{wo_w^* p_w^*})$

    $\mathscr{P}' \leftarrow \mathscr{P}' \cup \texttt{startAt}(T_w, S_w^*)$

  end for

  $\mathscr{P}' \leftarrow \mathscr{P}' \cup \{\sum_{p \in P} n_p < \sum_{p \in P} n_p^*\}$

  Get solution $\mathscr{S} = \{(\bar{S}_w, \bar{p}_w, \bar{o}_w)_{w \in W}, (\bar{n}_p)_{p \in P}\}$ by solving $\mathscr{P}'$ with CP Optimizer under time limit $\min(t_{base} + t_{inc}, T -$ elapsed time$)$

  if $\sum_{p \in P} \bar{n}_p < \sum_{p \in P} n_p^*$ **then**

    $\mathscr{S}^* \leftarrow \mathscr{S}$

    $\tau_{inc} \leftarrow 0$

    $\pi_{ratio} \leftarrow 100$

  else if $\mathscr{S}$ is empty **then**

    $\tau_{inc} \leftarrow \tau_{inc} + 10$

    $\pi_{ratio} \leftarrow \pi_{ratio} - 10$

  end if

end while

return $\mathscr{S}^*$

Table 7: CP Optimizer - LNS Comparison

Inst	$\|W\|$	CP optimizer			LNS		
		1min	15min	30min	1min	15min	30min
A	90	6	6	6	6	6	6
B	159	17	17	13	20	14	13
C	455	20	19	19	19	18	18
D	455	20	20	18	22	18	18
E	721	24	21	21	25	21	21
F	486	23	20	19	23	17	17
G	165	20	16	15	24	15	13

Table 8: LNS Improvement over CP Optimizer

Inst	$\|W\|$	1min	15min	30min
A	90	0%	0%	0%
B	159	-17.6%	17.6%	0%
C	455	5%	5.2%	5.2%
D	455	-10%	10%	0%
E	721	-4.2%	0%	0%
F	486	0%	15%	10.7%
G	165	20%	6.2%	13.3%

Table 9: LNS - SSG Heuristic Comparison

Inst	$\|W\|$	Heuristic	LNS	impr
A	90	6	6	0%
B	159	18	13	27.7%
C	455	18	18	0%
D	455	18	18	0%
E	721	22	21	4.5%
F	486	26	17	23%
G	165		13	∞

are denoted by letters from A to G and some of them are much more difficult to solve. In fact, they are real-based instances coming from final assembly lines of an aircraft manufacturer. They have between 90 to 721 tasks each.

In our experiment we ran both CP-solving algorithm (with CP Optimizer) and the LNS algorithm on the instances with three different time limits: 1 minute, 15 minutes and 30 minutes. Table 7 displays the objective value of the best solutions found by both algorithm within the time limits. In the first column, the number of task per instance is also displayed, in order to provide a hint on the instance complexity. Both methods provided feasible solutions for all the 6 instances. Also, we can note that the LNS performs poorly within the 1 minute time limit. This is not really surprising, since the LNS algorithm starts by exploring the neighborhood of the first feasible solution found by CP Optimizer using the CP model. However, we observed that the LNS algorithm often finds better quality solutions than the CP approach within greater time limits. Table 8 that displays the improvement in percent shows that when the time limit is 15 minutes or 30 minutes the LNS approach never performs worse than CP Optimizer. The largest improvement is obtained for a CPU time limit of 15 minutes with an improvement on 5 instances out of 7.

Since instances A to G are real instances from the manufacturer, we had also the opportunity to compare the LNS algorithm performance to the results from the current activity-oriented serial scheduling generation (SSG) heuristic deployed at the aircraft manufacturer, described in Section 2. This comparison is shown in Table 9. It can be seen that the LNS leads to better solutions in 4 out of the 6 instances. For the two where it came to the same solution (C, D), the LNS was able to prove the optimality, which cannot be proven by the heuristic. Moreover, for instance G, the heuristic is not capable of finding a solution while the LNS provided one.

Table 10: LNS Improvement over CP Optimizer for the Multi-mode Resource Investment Problem

Inst.	1 min	15 min	30 min
$MRIP_{30}$	5.27%	11.15%	0.82%
$MRIP_{50}$	5.61%	15.47%	1.14%
$MRIP_{100}$	10.58%	17.46%	3.40%

6.4. Further Experiments

In this section, we propose to go a step further in the experiments. In Gerhards (2020) the author proposes a Constraint Programming model to solve the Multi-mode resource investment Problem (MMRIP). Formally, this problem is very similar to the assembly line scheduling problem at stake here. The notable differences are, firstly, the presence of non-renewable resources, and secondly, the objective function, where we want to minimize the weighted sum of the maximums of the resources used. In the same way that we used our CP model as a black box for the LNS algorithm, this time we will use the CP model that the author proposes as a black box and compare the results obtained by LNS with the results of his CP model. We use the same sweep algorithm as the one introduced in the previous section. We test our algorithm on the same instances as those used by the author, taken from the RIPlib dataset. These instances are separated into three subsets, with 30, 50 and 100 tasks. Note that we removed from these instances those that were solved by CP Optimizer in less than a minute, so that the results more accurately highlight the comparison on the difficult instances in this dataset. Both methods were tested within 1, 15, and 30 minutes. The average improvement (in %) are displayed in Table 10.

Looking at the results, we can notice several things. For all sets of instances, the LNS algorithm improves very little the solutions returned by the CP model solution after 30 minutes, which is explicable by the fact that most of the instances are closed after this time. On the other hand, it is with a time limit of 15 minutes that the difference in the quality of the solutions is the most noticeable: in this time the CP model is not yet solved, whereas the LNS algorithm has been able to explore a more interesting subset of solutions. It can be noted that this difference increases with the size of the instances.

7. Conclusions & Further Research

In this work, we have identified a gap in the traditional mixed-integer programming formulations of scheduling problems and the modeling needs for the

scheduling of aeronautical assembly stations. We have chosen to extend the event-based formulations for our problem. To do so, we have dealt with minimal time lags, multi-mode scheduling and resource leveling objectives. These constraints had been seldom addressed on their own in the existing literature and no mixed-integer linear program have been found dealing with them together. Due to these features, the two new formulations we have developed are a contribution not only for the aeronautical industry but also for general scheduling.

The computational results from Section 4.3 have proven that the model is able to solve up to optimality small instances. As well as this, they have enabled as to make a comparative study between the two event-based formulations: SEE-M and OOE-M formulations. The results of these comparisons are coherent with the ones reported by Koné et al. (2011) for the single mode with only precedence constraints model.

However, in order to extend it to bigger instances, it is necessary to improve the solver performances. Both SEE-M and OOE-M formulations have a number of constraints that grows as $O(|E|^2|W|)$ where $|E|$ is the number of events and $|W|$ the number of work tasks to be scheduled. From the computational results we know that most of the instances can be solved with less events than the established upper bound of $|W|+1$ for SEE-M and $|W|$ for OOE-M. Therefore, the use of pre-processing to approximate the real needed number of events will lead to major performance improvements.

On the other hand, a Constraint Programming model has also been proposed. Directly solved via the CP Optimizer solver, it has had a promising performance even with bigger and real instances.

Remarking that the methods embedded inside the CP Optimizer solver, especially the LNS method described in Godard et al. (2005) are time oriented and globally aim at compacting the schedule, we proposed a new LNS method more adapted to the resource leveling objective. We used a fast sweep algorithm to compute the load peaks and a self-adaptive neighborhood to efficiently reschedule the tasks involved in such peaks and their predecessors.

The method outperforms CP Optimizer on the industrial instances for medium CPU time limits (15 and 30 minutes).

Moreover, it manages to decrease by more than 20% the resource levels reached by the heuristic currently used by the aircraft manufacturer on the industrial instances, potentially yielding substantial gains.

Finally, the proposed LNS method substantially improves the results of the CP model proposed by Gerhards (2020) on hard instances of a related multi-mode resource investment problem for short CPU time limits (1 and 15 minutes).

A future research direction are the associated filtering techniques to solve the problem under study. CP solution scheme encompasses also constraint propagation techniques that can be used as pre-processing to calculate the relevant number of events; this issue would lead to major performance improvements. A promising research direction is to propose a hybrid MILP/CP large neighborhood search heuristic, as the one proposed for the MISTA challenge 2013 on the multi-mode RCPSP (Artigues and Hébrard, 2013). Ergonomic constraints and objectives for operators could also be considered for the problem at hand as recently proposed in a related problem by Arkhipov et al. (2018).

Acknowledgements

This work has been partially funded by the French National Research Agency (ANR) project ANR-18-CE10-0007 "PER4MANCE".

The authors want to acknowledge the support provided by the Spanish Agencia Estatal de Investigación, through the research project with code RTI2018-094614-B-I00.

References

Alvarez-Valdés, R., Tamarit, J., 1993. The project scheduling polyhedron: Dimension, facets and lifting theorems. European Journal of Operational Research 67, 204–220.

Arkhipov, D., Battaïa, O., Cegarra, J., Lazarev, A., 2018. Operator assignment problem in aircraft assembly lines: a new planning approach taking into account economic and ergonomic constraints. Procedia CIRP 76, 63–66. 7th CIRP Conference on Assembly Technologies and Systems (CATS 2018).

Artigues, C., Demassey, S., Néron, E., 2010. Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications. Wiley Library.

Artigues, C., Hébrard, E., 2013. MIP relaxation and large neighborhood search for a multi-mode resource-constrained multi-project scheduling problem, in: 6th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA), Ghent, Belgium. pp. 814–819.

Artigues, C., Michelon, P., Reusser, S., 2003. Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research 149, 249–267.

Baptiste, P., Le Pape, C., Nuijten, W., 2001. Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems. International Series in Operations Research and Management Science, Kluwer.

Biele, A., Mönch, L., 2018. Hybrid approaches to optimize mixed-model assembly lines in low-volume manufacturing. Journal of Heuristics 24, 49–81.

Borreguero, T., Artigues, C., Álvaro García, Lopez, P., Ortega, M., 2015a. Multimode time-constrained scheduling problems with generalized temporal constraints and labor skills, in: 7th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2015), pp. 809–813.

Borreguero, T., Garcia, A., Ortega, M., 2015b. Scheduling in the aeronautical industry using a mixed integer linear problem formulation scheduling in the aeronautical industry using a mixed integer linear problem formulation. The Manufacturing Engineering Society International Conference, MESIC 2015 Procedia Engineering 132, 982–989.

Borreguero, T., Mas, F., Menéndez, J., Barreda, M., 2015c. Enhanced assembly line balancing and scheduling methodology for the aeronautical industry. Procedia engineering 132, 990–997.

Boysen, N., Fliedner, M., Scholl, A., 2009. Sequencing mixed model assembly lines survey classification and model critique. European Journal of Operational Research 192, 349–373.

Brucker, P., Drexl, A., Mohring, R.and Neumann, K., Pesch, E., 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112, 3–41.

Brucker, P., Knust, S., 2011. Complex Scheduling. GOR-Publications, Springer.

Cordeau, J.F., Laporte, G., Pasin, F., Ropke, S., 2010. Scheduling technicians and tasks in a telecommunications company. Journal of Scheduling 13, 393–409.

Gerhards, P., 2020. The multi-mode resource investment problem: a benchmark library and a computational study of lower and upper bounds. OR Spectrum 42, 901–933.

Godard, D., Laborie, P., Nuijten, W., 2005. Randomized large neighborhood search for cumulative scheduling., in: ICAPS, pp. 81–89.

Hartmann, S., Briskorn, D., 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207, 1–14.

Hemmelmayr, V.C., Cordeau, J.F., Crainic, T.G., 2012. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Computers & Operations Research 39, 3215–3228.

Kagermann, H., Wahlster, W., 2013. Recommendations for implementing the strategic initiative Industry 4.0: Securing the future of german manufacturing industry. Final report of the Industry 4.0 Working Group .

Koné, O., Artigues, C., Lopez, P., Mongeau, M., 2011. Event-based MILP models for resource-constrained project scheduling problems. Computers & Operations Research 38, 3–13.

Kreter, S., Schutt, A., Stuckey, P.J., 2017. Using constraint programming for solving RCPSP/max-cal. Constraints 22, 432–462.

Laborie, P., Godard, D., 2007. Self-adapting large neighborhood search: Application to single-mode scheduling problems. Proceedings MISTA-07, Paris 8.

Laborie, P., Rogerie, J., Shaw, P., Vilím, P., 2018. IBM ILOG CP Optimizer for scheduling. Constraints 23, 210–250.

Mas, F., Arista, R., Oliva, M., Hiebert, B., Gilkerson, I., Ríos, J., 2015. A review of PLM impact on US and EU aerospace industry. Procedia Engineering 132, 1053–1060.

Mas, F., Menéndez, J.L., Oliva, M., Servan, J., Arista, R., Del Valle, C., 2014. Design within complex environments: Collaborative engineering in the aerospace industry, in: Information System Development. Springer, pp. 197–205.

Nouri, N., Krichen, S., Ladhari, T., Fatimah, P., 2013. A discrete artificial bee colony algorithm for resource-constrained project scheduling problem, in: 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO 2013), pp. 1–6. doi:10.1109/ICMSAO.2013.6552557.

Palpant, M., Artigues, C., Michelon, P., 2004. Lssper: Solving the resource-constrained project scheduling problem with large neighbourhood search. Annals of Operations Research 131, 237–257.

Polo-Mejía, O., Artigues, C., Lopez, P., Basini, V., 2020. Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility. International Journal of Production Research 58, 7149–7166.

Pritsker, A.A.B., Watters, L.J., Wolfe, P.M., 1969. Multiproject scheduling with limited resources: A zero-one programming approach. Management Science 16, 93–108.

Schläpfer, R.C., Koch, M., Merkofer, P., 2014. Industry 4.0. Challenges and solutions for the digital transformation and use of exponential technologies. Technical Report. Deloitte. Zurich.

Schwindt, C., Zimmermann, J., 2015. Handbook on Project Management and Scheduling Vol.1. Springer.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems, in: Maher, M.J., Puget, J. (Eds.), Principles and Practice of Constraint Programming - CP98, 4th International Conference, Pisa, Italy, October 26-30, 1998, Proceedings, Springer. pp. 417–431.

Thomas, C., Schaus, P., 2018. Revisiting the self-adaptive large neighborhood search, in: International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Springer. pp. 557–566.

Wang, H., Li, T., Lin, D., 2010. Efficient genetic algorithm for resource-constrained project scheduling problem. Transactions of Tianjin University 16, 376–382.

Zapata, J., Hodge, B., Reklaitis, G., 2008. The multimode resource constrained multiproject scheduling problem: alternative formulations. AIChEJournal 54(8), 2101–2119.