Large neighborhood search for an aeronautical assembly line time-constrained scheduling problem with multiple modes and a resource leveling objective

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

Large neighborhood search for an aeronautical assembly line time-constrained scheduling problem with multiple modes and a resource leveling objective

Tamara Borreguero Sanchidrián[1,3], Tom Portoleau[2], Christian Artigues[2], Alvaro García Sánchez[3], Miguel Ortega Mier[3], Pierre Lopez[2]

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

Abstract

This paper deals with a scheduling problem arising at the tactical decision level in aeronautical assembly line. It has the structure of a challenging multi-mode resource-constrained project scheduling problem with incompatibility constraints, a resource leveling objective and also a large number of tasks. We first present a new event-based mixed-integer linear programming formulation and a standard constraint programming formulation of the problem. A large-neighborhood search approach based on the constraint programming model and tailored to the resource leveling objective is proposed. The approaches are tested and compared using industrial instances, yielding significant improvement compared to the heuristic currently used by the company. Moreover, the large-neighborhood search method significantly improves the method proposed in the literature on a related multi-mode resource investment problem when short CPU times are required.

Keywords: Aeronautical assembly line scheduling, multiple modes, resource leveling, mixed-integer linear programming, constraint 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 along with their levels of complexity and customization. As a result, aircraft manufacturers have now 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, Bortolini et al., 2021]. 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.

To fill this gap, this work addresses an NP-hard scheduling problem arising at the tactical decision level in the final assembly line by solving a multi-mode resource-constrained project scheduling problem (MMR-CPSP) with incompatibility and temporal constraints, a resource leveling objective and a large number of tasks.

A mixed-integer linear programming (MILP) formulation has been previously proposed for the considered problem by Borreguero-Sanchidrián et al. [2014], Borreguero et al. [2015a] and Borreguero et al. [2015c], extending the event-based formulation initially proposed for the standard resource-constrained project scheduling problem (RCPSP) [Koné et al., 2011] to tackle the additional industrial characteristics, but was only able to solve exactly very small instances.

In this paper, we propose a new MILP formulation and a constraint programming (CP) model. Based on the CP model, we propose a new large neighborhood search (LNS) heuristic with a neighborhood structure tailored to the resource leveling objective, with the aim of improving the performance of both exact methods and the heuristic approach currently used by an aircraft manufacturer on large-scale industrial instances.

The structure of the article is as follows: Section 2 presents the industrial context and a literature review. Section 3 provides a detailed problem description. Section 4 presents the new MILP formulations and a standard CP model. In Section 5, a novel LNS heuristic is presented. In Section 6, all approaches are compared, including a comparison with 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. The proposed LNS approach outperforms all other approaches on the industrial instances and also the method proposed by Gerhards [2020] on a resource investment problem. Finally, conclusions and further research directions are presented in Section 7.

2 Industrial context and literature review

The design and operation of an aeronautical assembly line is a complex problem that follows different decision steps, taking into account the decision time horizon. Within those steps, the main decisions regarding planning and scheduling process in aeronautical assembly lines follow the classical three-level hierarchy: strategical, tactical and operational [Buergin et al., 2018].

At the strategical level, the products to be manufactured are assigned to the different facilities within the supply chain. Moreover, the high level

design of each of the facilities is made. In most cases, the assembly of the major airframes as well as the final assembly are organized as flow shops. They are built as moving or pulse lines, where each product has to go through all the stations following a fixed path. Some of the aeronautical assembly tasks require complex jigs and robots. Therefore, a preliminary line balancing is performed in this early stage in order to allocate major tasks to stations and, as a result, be able to design each station in terms of jigs and robots. Assembly line balancing techniques, where jobs have to be optimally assigned to stations, have been applied to aeronautical assembly lines in the literature. 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. A recent survey on assembly line balancing can be found in Boysen et al. [2022].

In the mid term, tactical decisions include capacity planning and master production scheduling. In the aeronautical industry, aircraft assembly is a labor-intensive process that requires highly qualified operators [Mas et al., 2016]. In consequence, the teams need to be sized early enough to ensure their availability. To do so, the preliminary line balancing is reviewed. Afterwards, a preliminary scheduling of the tasks within a station is made. At this mid term stage, the line cycle time is also assessed. Sometimes, there can be a tradeoff between different cycle times, related to a set of delivery rates. Shorter cycle times need in most cases more resources per station. Nevertheless, there is no linear relationship between those dimensions and several experiments may be needed to assess the impact of cycle time reduction in the team size. In consequence, the purpose of this scheduling is to calculate the minimum number of human resources needed to ensure the station tasks can be finished on time [Borreguero, 2020].

In the short term, an operational, more detailed, schedule is established per station. This new scheduling process deals with a more detailed modeling of the resource availability, including shift planning, considering individualization of operators and precise modeling of their skills, which was only approximated at the previous stage. On top of this, inflows and other contingencies, such as material shortage or defects, must be taken into account. Moreover, the objective at this stage is to minimize the tardiness, given the available resources. Related studies for integrated project scheduling and shift planning have been undertaken by Maenhout and Vanhoucke [2013].

Our paper deals with the scheduling problem defined at the tactical level where tasks have already been assigned to stations, as stated in Borreguero et al. [2015b], with the objective to find a solution that optimizes the operator usage for a fixed cycle time. In this problem, the operators are not individualized and precise modeling of their skills is not possible. The skills required by the tasks are aggregated in the concept of resource profile, as explained in Section 3.1. The operational/short term scheduling problem is not addressed. We refer to Kolisch [2000] for a survey of make-to-order assembly lines, where the problem of operations scheduling for aircraft assembly lines is described in Section 2.5.4. We are dealing with a project scheduling problem of one station consisting of a set of

activities of known duration and resource profile requests, that must be executed under incompatibility and precedence constraints. Given a time limit for the project duration, the objective is to find the schedule that consumes the least amount of resources. This turns the problem into a time-constrained variant of the MMRCPSP with resource leveling objective.

There have been a wide range of studies on both heuristic and metaheuristic methods for solving the RCPSP, as well as different MILP models [Artigues et al., 2010, 2015, Wang et al., 2010, Hartmann and Briskorn, 2010, Brucker and Knust, 2011, 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 advantage of efficiently solving instances with large time horizon. They concluded that event-based formulations outperformed the discrete time MILP models for large scheduling horizons and outperformed also the continuous time flow-based formulations for highly "cumulative" instances (where many activities can be scheduled in parallel), 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 interest to enhance these formulations so that they can be used within more practical RCPSP contexts. As mentioned above, an event-based MILP formulation has been previously proposed for the problem studied in this paper [Borreguero-Sanchidrián et al., 2014, Borreguero et al., 2015a,c] but was only able to solve exactly problems instances with up to 11 tasks.

Constraint Programming (CP), on the other hand, provides both 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]. In consequence, CP is a suitable technique for solving variants of RCPSP including those with features such as general temporal constraints, calendar constraints [Kreter et al., 2017] and multi-skill operators [Polo Mejía et al., 2020]. In fact, a CP 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. A CP model has been also proposed for the closely related resource investment problem by Gerhards [2020].

Large Neighborhood Search (LNS) heuristics enable dealing with large-scale industrial instances, while benefiting from the power of MILP or CP solvers. This technique, based on the iterative solving of a subproblem where a part of a current solution of the global problem is fixed while the rest is freed, was originally introduced in Shaw [1998] and turned out to perform well on several problems, such 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 algorithms 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 has never been applied to the MMRCPSP with resource leveling objectives such as in our case.

In the MMRCPSP, each task has a set of execution modes. A mode defines a duration of the task and a number of units required on each resource. The logic is that a mode with a shorter duration requires more resources than a mode with a longer duration.

The literature on the MMRCPSP is quite rich with regard to the minimization of the total duration of the schedule (makespan) [Drexl and Gruenewald, 1993, De Reyck and Herroelen, 1999, Weglarz et al., 2011, Coelho and Vanhoucke, 2011]. A fairly recent overview of (MM)RCPSP models and solution approaches can be found in Schwindt and Zimmermann [2015]. In relation to the industrial problem in the aeronautical assembly considered in this paper, we are interested in the problem of smoothing the resources given a fixed total duration. In the literature on the MMRCPSP, this is referred to as resource leveling [Demeulemeester and Herroelen, 2002, Schwindt and Zimmermann, 2015, Coughlan et al., 2015]. Resource leveling consists in minimizing the maximum quantity of each resource used at each time point (the resource peak). When acquisition costs are associated with resources, we speak of the resource investment problem [Gerhards, 2020] or the resource-availability cost problem [Demeulemeester, 1995].

Another special feature of our problem is the scheduling of workers in a constricted area. This had been previously studied by Brimberg et al. [1996].

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 in AIRBUS. It uses an activity-oriented serial scheduling generation heuristic [Kolisch, 1996, Borreguero et al., 2015b]. Its priority rules are focused on the minimum start time per task: select the task with the smallest latest start time and, in case of a tie, select the task with the smallest earliest start time. This heuristic is good at providing feasible solutions fast enough, given a station's cycle time and a resource availability, but can lead to significantly sub-optimal solutions.

Given this literature review and the industrial context, this paper investigates how to improve the previously proposed event-based MILP formulation and how to design a constraint programming-based LNS method tailored to the resource leveling problem, with a double objective: improving the heuristic used by AIRBUS on the industrial problem and competing with a state-of-the-art approach on the resource-investment problem.

3 Problem description

As stated above, we are dealing with the scheduling of an assembly station belonging to the last step of the tactical decisions. It is used to size

the teams allocated to each station. Therefore, some of the operators' characteristics, such as skills or shifts, need not be modeled in detail. The following are the main features of the problem we have tackled.

3.1 Operator profiles and objective function

Aeronautical assembly requires highly-qualified operators. Often, some of the tasks need to be done by workers with a specific qualification. These qualifications are managed creating profiles that group qualifications. Each operator is assigned to a profile depending on his qualifications. Operators work in teams where all the operators have the same profile. Consequently, we have a set of resources $\mathcal{R}$, each corresponding to an operator profile, without a precise identification of operators.

Within a real team, there are operators of different skill levels. For example a typical profile is "mechanic" or "electrician" while skills correspond to detailed competencies such as "rivet drilling", "sealant placing", etc. Skills also include levels of ability since some tasks must be performed by an experienced riveter. The assignment of tasks to operators depending on their precise skills are managed by the team leader during the operational scheduling, which is not addressed in this paper. Therefore, for the intermediate tactical level, we model the operators as belonging to a single profile, independently from their skill level.

Also, the operator teams work in a single station during the whole cycle time. In consequence, the objective of the scheduling is to minimize the peak demand of operators, as the sum of the peak demand per profile, which is the minimum team size resulting decided at the tactical level.

3.2 Tasks characteristics

The set of all tasks to be performed at the considered station is denoted V and $\mathcal{R}_j$ denotes the set of resources able to perform task $j \in V$, i.e. one and only one resource must be selected in set $\mathcal{R}_j$ to perform task j.

For each task $j \in V$, there is a range R_j of possible demands (number of resource units/operators). Increasing the number of resource units assigned to a task reduces the task duration in a non-linear task-specific manner. Hence, parameter p_{jqk} gives the duration of task $j \in V$ when performed by $q \in R_j$ units of resource $k \in \mathcal{R}_j$. By reference to the standard MMRCPSP, this corresponds to the restriction where a mode consists of a single resource and a number of resource units.

Stations can be of very different sizes. However, even the smallest stations are big enough for several tasks to be performed in parallel. Nevertheless, it is usually the case that several tasks are to be done near to each other, in a way that some subset of tasks cannot be executed at the same time due to space constraints. As a result, stations are divided into 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 the scheduling problem. For each task and mode, a required amount of space per area is defined as the number of resource units $q \in R_j$ used by each task j in the considered mode. The set of areas is denoted as A, a_j

denotes the area where task j is performed and C_a the number of available resource units in area a. Finally, task preemption is not allowed.

3.3 Time constraints between tasks

The constraints between tasks can be of different nature. All the time constraints are independent from the mode in which a task is executed.

The most common case of time constraints is that of finish-start precedence constraints: when a task cannot be started until a previous one has been finished. For example: wire harnesses cannot be installed until the brackets 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. We denote as E the set of finish-start precedence constraints such that $(i, j) \in E$ means that task j cannot start before the completion of task i.

Another kind of time constraints are incompatibility constraints: this means 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 few 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. We denote as D the set of incompatibility constraints, such that $\{i, j\} \in D$ means that tasks i and j cannot overlap.

Finally, maximum time lags also occur. For example, electrical continuity tests have to be performed just after the cleaning of the surfaces and bonding points need to be protected with sealant almost immediately after the test. The set of maximum time lags constraints is denoted by L, such that $(i, j) \in L$ means that j cannot start after the completion of i plus a fixed time lag Δ.

All the tasks must be completed within the station's cycle time. Hence we consider a fixed time horizon (set of time points) T, which amounts to finding a schedule of makespan lower than $|T|$.

3.4 Synthesis of simplifying assumptions

This problem, as it has been described in the previous subsections, corresponds to the tactical decision level for an aeronautical assembly line. Therefore, it includes some simplifying assumptions that may be taken into account before its extension to other use cases:

1. Operators are assigned to a single profile. At the tactical level, the operators are not individualized and the precise modeling of their skills is not taken into account. This could not be done at that stage because the detailed information on the operators that will be available is not known. However, at the operational level, this assumption may be too restrictive, as operators typically have a set of skills instead of a single one.
2. Resource capacity is modeled as constant through the cycle time. This can be a major drawback to consider this assumption at the opera-

tional level, where we have to deal with different capacities for the resources, e.g., in early or late shifts or due to operator absence. As a consequence, when the resource capacity decreases, some on-going task may have to be interrupted and resumed at a later time when the capacity becomes sufficient again. However, at the tactical level we do not have the detail of the operator availabilities and, also, as explained in Section 2, the major objective at the considered level is to find a tradeoff between the cycle time and the maximum resource usage, which justifies a constant availability.

3. Objective function: minimizing the total number of operators. This objective function can indirectly define a cost function, as the aeronautical industry is human resource intensive. In the assembly line problem, we assume that all the profiles have the same cost on average. This is not a real limitation as a weighted objective function can easily be used, as for the resource investment problem in Section 6.5.
4. Only one operator type is required to perform a task. In our model, one mode of our MMRCPSP is defined by a single operator type and its required amount. This means that a task cannot be executed by a set of operators from different profiles. This assumption is driven from the fact that, in the use case we took as a reference, operators are organized in teams driven by profiles. Each team, even if working in a single station, has a set of tasks to be performed independently from the other teams. Therefore, tasks are assigned to a single team and in consequence to a profile, and mixed execution modes are not allowed. An extension to more general modes is not an issue; for example in Section 6.5, we solve a problem where each task requires several resources, including non-renewable ones.
5. Space is modeled as a renewable resource. The occupied space becomes available again in its initial capacity as soon as freed by the operators working on tasks that occupy it. An extension of the model could take into account the space availability which may change once the tasks are completed. Nevertheless, we can assume a uniform space availability per station and incremental changes between stations.
6. Tasks are separated by a fixed time lag. This assumption has been data-set driven. After a study of a wide range of real data sets (other than the ones used in the experimentation), maximal time lags used had always a zero finish-start time lag. In practice, this is because this is used for bonding points protection, which should be done as soon as possible, and also for other cases where they want to ensure continuity in the scheduling of certain tasks. Nevertheless, this assumption has little impact in the models and could be easily generalized to a precedence-dependent time lag.

3.5 A conceptual formulation

Overall, solving the considered MMRCPSP aims at assigning a start time S_j, a resource k_j and a number of units q_j to each task $j \in V$ so as

to minimize the sum of peak unit demand per resource, as stated by the following conceptual formulation. Table 1 recalls the MMRCPSP sets and parameters, while Table 2 describes the variables used in the conceptual formulation.

Table 1: Sets and parameters for the MMRCPSP

	Sets
V	set of tasks
$\mathcal{R}$	set of resources
T	set of time periods
$\mathcal{R}_j$	set of possible resources required by task $j \in V$
R_j	set of possible numbers of resource units required by $j \in V$
A	set of areas
E	set of finish-start precedence constraints
L	set of finish-start maximum time lags
D	set of incompatibility constraints
	Parameters
$\|T\|$	length of time horizon T
p_{jqk}	duration of task $j \in V$ when assigned to $q \in \mathcal{R}_j$ units of resource $k \in \mathcal{R}_j$
Δ	maximum time lag
a_j	area required by task $j \in V$
C_a	capacity of area $a \in A$

Table 2: Decision variables used by the MMRCPSP conceptual formulation

$r(k,t)$	number of resource $k \in \mathcal{R}$ units required by all tasks at time $t \in T$
$V(t)$	set of tasks in process at time $t \in T$
S_j	start time of task $j \in V$
k_j	resource assigned to task $j \in V$
q_j	number of resource units assigned to task $j \in V$

$$\min \sum_{k \in \mathcal{R}} \max_{t \in T} r(k,t) \tag{1}$$

$$V(t) = \{j \in V, t \in [S_j, S_j + p_{jq_jk_j} - 1]\} \qquad t \in T \tag{2}$$

$$r(k,t) = \sum_{j \in V(t), k_j = k} q_j \qquad \forall k \in \mathcal{R}, t \in T \tag{3}$$

$$S_j \geq S_i + p_{iq_ik_i} \qquad (i,j) \in E \tag{4}$$

$$S_j \leq S_i + p_{iq_ik_i} + \Delta \qquad (i,j) \in L \tag{5}$$

$$S_j \geq S_i + p_{iq_ik_i} \vee S_i \geq S_j + p_{jq_jk_j} \qquad \{i,j\} \in D \tag{6}$$

$$\sum_{j \in V(t), a_j = a} q_j \leq C_a \qquad a \in A \tag{7}$$

$$S_j \leq |T| \qquad j \in V \tag{8}$$

$$S_j + p_{jq_jk_i} \leq |T| \qquad i \in V \tag{9}$$

$$q_j \in R_j \qquad i \in V \tag{10}$$

$$k_j \in \mathcal{R}_j \qquad i \in V \tag{11}$$

The objective function (1) minimizes the sum of maximum resource usage over the time horizon. The set of tasks $V(t)$ in process at time period t is given by expression (2). The resource k total usage at time period t is given by expression (3). Constraints (4) define the finish-start precedence constraints while constraints (5) are the maximum time lag constraints. Disjunctive constraints are expressed by disjunctions (6). Area capacity constraints are given by (7). The definition domains of start and end variables are given by (8) and (9), respectively while the domain of variables q_j and k_j is given by constraints (10) and (11).

The problem is a special case of the MMRCPSP with minimum and maximum time lags with the following restrictions: a mode is defined by a single resource and its required amount, the minimum time lags are standard finish-start precedence constraints and the maximum time lags are of finish/start type with a fixed offset, Δ. Although these restrictions – that reflect the considered industrial case – make the problem significantly different (and simpler) than the general MMRCPSP with minimum and maximum time lags, the problem remains NP-hard as its decision variant admits the decision variant of the single-resource RCPSP as a special case. Furthermore, the methods proposed in the paper can be easily extended to the general case, as shown by the computational experiments we present for the resource investment problem in Section 6.5.

4 Exact methods

This section presents two exact methods for the problem : one based on a new MILP formulation and the second one based on a standard CP formulation.

4.1 MILP formulation

Several MILP formulations are proposed in Koné et al. [2011]. They can be roughly divided into two categories: the discrete time models (time-indexed formulations) and the continuous time models (event-based and sequencing formulations). As the target industrial instances have a very large time horizon (see Section 6) using time-indexed models is hopeless.

Borreguero-Sanchidrián et al. [2014], Borreguero et al. [2015a] and Borreguero et al. [2015c] have proposed an event-based formulation for the problem, extending the start/end event-based formulation proposed 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-indices in order to deal with the multiple modes per task. A set of events $\mathcal{E}$, numbered from 0 to $n = |\mathcal{E}| - 1$, models the significant time points. Two sets of binary variables, x_{jeqk} and y_{jeqk}, were used to define the start and end events of each task, as $x_{jeqk} = 1$ ($y_{jeqk} = 1$) if task $j \in V$ starts (ends) at event $e \in \mathcal{E}$ using q units of resource $k \in \mathcal{R}$. Furthermore a binary variable α_{ij} was needed for each pair $\{i, j\} \in D$. We will refer to this formulation as SEE-M.

We propose a new event-based formulation based on the on-off formulation presented in Koné et al. [2011] that needs a single set of binary variables z_{jeqk}, whose value is 1 if task j is active between event e and event $e + 1$ using q units of resource k and 0 otherwise. Another set of binary variables, β_{jqk} is used to choose the mode (q, k) in which a task j is performed. There is no need for variables α_{ij} for the disjunctive constraints. Continuous variable t_e gives the time of event $e \in \mathcal{E}$. Continuous variables r_k and S_j model the peak usage of resource k and the start time of task $j \in V$, respectively. As the resource profile only increases when a task starts, we only need to define one event per task, hence $n = |V| - 1$. For ease of notation, we will denote $\mathcal{M}_j$ the set of modes given by $R_j \times \mathcal{R}_j$. We will refer to this as the OOE-M formulation. We recall that Table 3 synthesizes the additional sets and variable parameters used by the OOE-M formulation, given as follows:

Table 3: Additional sets and decision variables for the OOE-M formulation

	Sets
$\mathcal{E}$	set of events
$\mathcal{M}_j$	set of modes for task $j \in V$ ($\mathcal{M}_j = R_j \times \mathcal{R}_j$)
	Decision variables
S_j	start time of task $j \in V$
z_{jeqk}	binary variables equal to 1 if task $j \in V$ is active at event $e \in \mathcal{E}$ using $q \in \cup_{j \in V / k \in \mathcal{R}_j} R_j$ unit of resource $k \in \mathcal{R}$
β_{jqk}	binary variable equal to 1 is task $j \in V$ uses $q \in R_j$ units of resource $k \in \mathcal{R}_j$

$$\min \sum_{k \in \mathcal{R}} r_k \tag{12}$$

$$t_0 = 0 \tag{13}$$

$$\sum_{j \in V} \sum_{(q,k) \in \mathcal{M}_j} z_{j0qk} \geq 1 \tag{14}$$

$$t_{e+1} - t_e \geq 0 \quad e \in \mathcal{E} \setminus \{n\} \tag{15}$$

$$\sum_{e \in \mathcal{E}} \sum_{(q,k) \in \mathcal{M}_j} z_{jeqk} \geq 1 \quad j \in V \tag{16}$$

$$\sum_{(q,k) \in \mathcal{M}_j} \beta_{jqk} = 1 \quad j \in V \tag{17}$$

$$z_{jeqk} \leq \beta_{jqk} \quad j \in V, e \in \mathcal{E}, \; (q,k) \in \mathcal{M}_j \tag{18}$$

$$\sum_{e'=0}^{e-1} \sum_{(q,k) \in \mathcal{M}_j} z_{je'qk} - e \left(1 - \sum_{(q,k) \in \mathcal{M}_j} (z_{jeqk} - z_{je-1qk}) \right) \leq 0 \quad j \in V, e \in \mathcal{E} \tag{19}$$

$$\sum_{e'=e}^{n-1} \sum_{(q,k) \in \mathcal{M}_j} z_{je'qk} - (n-e) \left(1 + \sum_{(q,k) \in \mathcal{M}_j} (z_{jeqk} - z_{je-1qk}) \right) \leq 0 \quad j \in V, e \in \mathcal{E} \tag{20}$$

$$t_f - t_e - \sum_{(q,k) \in \mathcal{M}_j} p_{jqk} (z_{jeqk} - z_{je-1qk} - z_{jfqk} + z_{jf-1qk}) \geq - \sum_{(q,k) \in \mathcal{M}_j} p_{jqk} \beta_{jqk} \quad f, e \in \mathcal{E}, \; f > e, j \in V \tag{21}$$

$$|T| - t_e - \sum_{(q,k) \in \mathcal{M}_j} p_{jqk} (z_{jeqk} - z_{je-1qk}) \geq 0 \quad j \in V, \; e \in \mathcal{E} \tag{22}$$

$$\sum_{(q,k) \in \mathcal{M}_i} z_{ieqk} + \sum_{e'=0}^{e} \sum_{(q,k) \in \mathcal{M}_j} z_{je'qk} - (e-1)(1 - \sum_{qk} z_{ieqk}) \leq 1 \quad e \in \mathcal{E}, (i,j) \in E \tag{23}$$

$$S_j \geq t_e - |T|(1 + z_{je-1qk} - z_{jeqk}) \quad e \in \mathcal{E}, \; j \in V \tag{24}$$

$$S_j \leq t_e + |T|(1 + z_{je-1qk} - z_{jeqk}) \quad e \in \mathcal{E}, \; j \in V \tag{25}$$

$$S_j - (S_i + \sum_{(q,k) \in \mathcal{M}_i} p_{iqk}) \leq \Delta \quad (i,j) \in L \tag{26}$$

$$\sum_{q \in R_i, k \in \mathcal{R}_i} z_{ieqk} + \sum_{(q,k) \in \mathcal{M}_j} z_{jeqk} \leq 1 \quad (i,j) \in D \tag{27}$$

$$\sum_{j \in V} \sum_{q \in R_j} q z_{jeqk} \leq r_k \quad k \in \mathcal{R}, e \in \mathcal{E} \tag{28}$$

$$\sum_{j \in V, a_j = a} \sum_{(q,k) \in \mathcal{M}_j} q z_{jeqk} \leq C_a \quad a \in A, e \in \mathcal{E} \tag{29}$$

$$z_{jeqk} \in \{0,1\} \quad e \in \mathcal{E}, j \in V, \; k \in \mathcal{R}_j, q \in R_j \tag{30}$$

The objective function (12) is to minimize the total project cost. The first event on the project starts at $t = 0$ as stated by Constraints (13) and at least one task must be active after this event as per Constraint (14). Constraints (15) sort the events.

Constraints (16) state that each task must be active at least after one of the events in order to ensure the scheduling of all the tasks. Variables β_{jqk} select the mode in which each task will be performed. One and only one of the variables β_{jqk} can be set to 1 per task, Constraint (17), and the tasks can only be performed on the selected mode, as per Constraint (18).

Constraints (19) to (22) are based on the three values than can take the difference $z_{jeqk} - z_{je-1qk}$ and enforce the on-off behavior of the variables:

- $z_{jeqk} - z_{je-1qk} = 1$; when e is the first event after which j is active, so $z_{jeqk} = 1$ and $z_{je-1qk} = 0$.
- $z_{jeqk} - z_{je-1qk} = -1$; when $e - 1$ is the last event after which j is active, so $z_{jeqk} = 0$ and $z_{je-1qk} = 1$.
- $z_{jeqk} - z_{je-1qk} = 0$; otherwise.

Constraints (19) and (20) refer to the continuous processing of each task: by Constraint (19) if task j begins after event e, then it cannot be processed before $e-1$. Similarly, by Constraint (20) if task j ends at event e then j 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 j is active) and the end event (the last event after which j is active).

The time difference between j's start event and its end event must be at least the task's processing time (21) and none of the tasks can end after the station cycle time, see (22).

Regular precedence constraints are Constraints (23), as if i must precede j, then it must start at an event after which i is no more active. Maximum time lags are expressed by Constraints (24) to (26). Constraints (24) and (25) define the start time of a task. These constraints will only be calculated for the tasks involved in maximum time lag constraints. Together with them, Constraints (26) limit the time between the end of a task and its successor's start time. Incompatibility constraints are Constraints (27), as two incompatible 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: Constraints (28) for the quantity of operators per type and Constraints (29) 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.

Appendix A presents a detailed computational comparison of the proposed MILP formulations. As it will be shown in the computational experiment section, MILP formulations are however not able to compete with the constraint programming approach described in the next section.

4.2 Constraint programming

Constraint Programming (CP) has been proven to be an efficient method on several combinatorial optimization problems, especially scheduling problems [Baptiste et al., 2001].

For modeling and solving, we used the CP Optimizer constraint-based scheduling library [Laborie et al., 2018]. More precisely, we used the basic modeling elements available in CP Optimizer, which are standard scheduling constraints. For the solving part, we simply run the default solver solution search procedure. We refer to Laborie et al. [2018] for a more detailed definition of these elements. We provide a short description of the CP Optimizer language elements we use for the readers that are not familiar with CP and we show how these elements are used to model our problem.

First, to model the objective function, we introduce a integer decision variable r_k for each resource $k \in \mathcal{R}$. This decision variable represents the maximum number of units of resource k required by the tasks. The objective function is then directly written as follows:

$$\min \sum_{k \in \mathcal{R}} r_k \tag{35}$$

The basic modeling element is the `interval` decision variable used to model a task. This decision encapsulates the start time, duration and end time decision variables. To model a task *task* with a release date *rel*, a deadline *due* and a duration *dur* one has just to write:

`dvar interval` *task* `in` *rel..due* `size` *dur*

Thus in our case, a task $j \in V$ is modeled as an `interval` decision variable T_j with a release date 0, a due date $|T|$ and an unspecified duration as the tasks have multiple modes:

$$\texttt{dvar interval } T_j \texttt{ in } 0..|T| \quad \forall j \in V \tag{36}$$

An `interval` decision variable can be declared `optional`, which means that the solver will actually assign values to the start, duration and end variables of this only if this is relevant for the optimization. An assigned `optional interval` decision variable is said to be `present`, otherwise it is `absent`. `optional interval` decision variables are used to model tasks with multiple modes in the sense that a mode can be modeled as an optional task with fixed resource requirements and duration. In the following declaration, `optional` interval variable T_{jqk} models the mode of task $j \in V$ when it is assigned to q units of resource k:

$$\texttt{dvar interval optional } T_{jqk} \texttt{ in } 0..|T| \texttt{ size } p_{jqk} \qquad \forall i \in V, (q,k) \in \mathcal{M}_j \tag{37}$$

An `interval` decision variable can be also declared as an `alternative` among different `optional interval` decision variables. In this case, one and only one of the `optional interval` decision variables will be present

and synchronized with the `alternative interval` decision variables. In our case, we declare task T_j as an `alternative` between tasks T_{jqk}:

$$\texttt{alternative}(T_j, (T_{jqk})_{(q,k)\in\mathcal{M}_j}) \quad \forall j \in V \tag{38}$$

Precedence constraints can be declared between `interval` decision variables, either in the form of classical finish-start precedence constraints or generalized precedence constraints with time lags. We use here the `endBeforeStart`($task1, task2$) constraint that enforces the standard finish-start precedence constraint between $task1$ and $task2$:

$$\texttt{endBeforeStart}(T_i, T_j) \quad \forall (i,j) \in E \tag{39}$$

We use also the `startBeforeEnd`($task1, task2$,δ) that sets a minimum (possibly negative) distance δ between the end of $task2$ and the start of $task1$, to model the maximum time lag constraints:

$$\texttt{startBeforeEnd}(T_j, T_i, -\Delta) \quad \forall (i,j) \in L \tag{40}$$

In CP Optimizer, the task resource usage is modeled by associating a resource usage function to an `interval` decision variable. The function will be equal to 0 outside of the task execution interval and equal to the number of resource units from the start to the end of the task. such a function is named a `pulse` function and the syntax to associate such a function to an `interval` decision variable $task-$ with a number of units q is `pulse`($task, q$). A global resource consumption function for a given resource can be obtained by summing all the individual task `pulse` functions. The resource capacity is enforced by declaring that this sum must stay lower than the capacity. We define here a sum per area and per resource and we associate the `pulse` function to the `optional interval` variables T_{jqk} that represent the modes. In case the `interval` variables T_{jqk} is absent, the `pulse` function remains equal to 0.

For each resource $k \in \mathcal{R}$, recall that decision variable r_k is used to model the maximal required number of units. It follows that the resource constraints are written as follows:

$$\sum_{j\in V} \sum_{q\in R_j} \texttt{pulse}(T_{jqk}, q) \leq r_k \quad \forall k \in \mathcal{R} \tag{41}$$

The capacity constraints for areas are written in a similar way:

$$\sum_{j\in V | a_j = a} \sum_{(q,k)\in\mathcal{M}_j} \texttt{pulse}(T_{jqk}, q) \leq C_a \quad \forall a \in A \tag{42}$$

The `noOverlap` constraint is used to indicate that two `interval` decision variables can not overlap, which allows to directly model the incompatibility constraints:

$$\texttt{noOverlap}(T_i, T_j) \quad \forall \{i,j\} \in D \tag{43}$$

The obtained CP model (35–43) is similar to the one used in Gerhards [2020]. Is serves as the core element of the LNS method proposed in the next section.

5 Large neighborhood search

5.1 The large neighborhood search technique

In the previous sections, we introduced MILP and CP formulations with the aim of solving exactly our problem. The experimental results reported in Section 6.2 will show that compared to the MILP approach, the CP Optimizer model gives much better results and manages to solve some instances with up to 100 tasks to optimality. These good results argue in favor of exploiting the CP approach inside a heuristic to solve larger problems. On that purpose, the LNS technique appears as a technique of choice. As stated in the introduction, it benefits from the power of the CP solver to explore a large-sized neighborhood. Furthermore, the LNS method has not been tailored yet on the resource leveling objective, which constitutes a methodological challenge.

In this section, we propose 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 $\mathcal{S}$ of the problem $\mathcal{P}$.

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

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

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

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

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

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 (1) so that its neighborhood contains better solutions; in other words, how to evaluate the criticality of a solution part?

5.2 LNS for the aircraft assembly line scheduling problem

As mentioned in the 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 thus seek to identify the set of tasks that are involved in the maximum resource peaks. Consider a solution $\mathcal{S}$ where each task $j \in V$ has start time $\bar{S}_j \in [0, |T|]$, a number of assigned units $\bar{q}_j \in R_j$ for resource $\bar{k}_j \in \mathcal{R}_j$, and a maximal usage $\bar{r}_k$ for each resource $k \in \mathcal{R}$. The set of peak tasks is the set of all critical sets as defined below:

Definition 1 *A critical set $\tilde{V}$ is a set of overlapping tasks that reaches the maximum number of operators for at least one resource $k \in \mathcal{R}$. More formally: $\exists t \in [0, |T|]$, $\exists k \in \mathcal{R}$, $\forall j \in \tilde{V}$, $\bar{k}_j = k$, $\bar{S}_j \leq t < \bar{S}_j + p_{j\bar{q}_j\bar{p}_j}$ and $\sum_{j \in \tilde{V}} \bar{q}_j = \bar{r}_k$.*

In fact, the resource usage only changes at the beginning or the end of a task. Let $\mathcal{T}$ denote the set of different start and end times 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 $\mathcal{T}$. Algorithm 1 describes the sweep algorithm that computes the union $\mathcal{C}$ of all peak task sets in $\mathcal{O}|V|^2|\mathcal{R}|$ time.

Algorithm 1 The `sweep` algorithm for peak task computation

Require: A problem $\mathcal{P}$ and a solution $\mathcal{S} = \{(\bar{S}_j, \bar{q}_j, \bar{k}_j)_{j \in V}, (\bar{n}_k)_{k \in \mathcal{R}}\}$
  $\mathcal{C} \leftarrow \emptyset$
  $\mathcal{T} \leftarrow \{\bar{S}_j \mid j \in V\} \cup \{\bar{S}_j + p_{jq_jk_j} \mid j \in V\}$
  for $k \in \mathcal{R}$ **do**
    for $t \in \mathcal{T}$ **do**
      $\tilde{V} \leftarrow \emptyset$; $cons \leftarrow 0$
      for $j \in V$ **do**
        if $\bar{k}_j = k$ **and** $\bar{S}_j \leq t < \bar{S}_j + p_{j\bar{q}_j\bar{k}_j}$ **then**
          $\tilde{V} \leftarrow \tilde{V} \cup \{j\}$; $cons \leftarrow cons + \bar{q}_j$
        end if
      end for
      if $cons = \bar{r}_k$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \tilde{V}$
      end if
    end for
  end for
  return $\mathcal{C}$

In order to generate a high quality neighborhood, we let free (Step 1 of the general LNS scheme) 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 are direct predecessors of them by a precedence constraint, and we fix the others.

We then solve this new problem (Step 2 of the LNS scheme) 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 principle originally proposed by Palpant et al. [2004]. To be more specific, each time the solver is unable to find a solution, we fix $\nu\%$ less activities and add μ seconds to the maximum solving time. These values were determined empirically using the benchmark instances (see Section 6).

In our implementation, we use CP Optimizer as a black box to solve different generated subproblems using the constraint programming model described in Section 4.2. Algorithm 2 provides the pseudo-code of our implementation of the LNS method for the aircraft assembly line scheduling problem.

Note that `presenceOf`(T_{jqk}) in the CP Optimizer language is a constraint that enforces the presence of the optional task T_{jqk}, while `startAt`(T_j, t) is a constraint that fixes the start time of task T_j to value t. These two constraints are used to fix the modes and the start times of the tasks in $V \setminus V'$ while the tasks in V' 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. In the algorithm, π_{ratio} gives the percentage of peak tasks and their predecessors that are taken to be included in the subproblem. This number is bounded from below by a limit $\pi^{\min}_{ratio}$. τ_{base} is the base time limit devoted to the LNS subproblem solving while τ_{inc} is the additional time that is increased by the increment μ at each iteration where no solution is found for the subproblem.

6 Computational experiments

Experiments were performed on a PC DELL Inspiron 1525 Intel Core2 Duo CPU, T5550 @ 1.83 GHz and 3.0 GB RAM. We first present the instances used for the experiments in Section 6.1. Then, a comparison between MILP and CP approaches is carried out in Section 6.2. The proposed LNS method is then compared to the CP approach and the heuristic used in the company in Section 6.3. The influence of the cycle time in the comparison between LNS and CP is studied in Section 6.4. Finally, a comparison between the LNS approach and a state-of-the-art method on the multi-mode investment problem is detailed in Section 6.5. The code and data instances are available at `https://gitlab.laas.fr/roc/christian-artigues/lns-mmrcpsp`.

6.1 Assembly line instances

On the real final assembly line, the cycle time varies from 14 to 25 working days. There are between two and seven operator profiles per station. Sample profiles are: mechanical technician, electrical technician, fluid systems technician, inspector and test specialist. As for the working areas, there can be defined up to 5 different areas per station.

Precedence constraints are the most frequent. There are normally a small percentage of jobs that do not have any kind of precedence relationship with the others. Tasks are usually organized on several groups (1–10)

Algorithm 2 LNS for the aircraft assembly line scheduling problem

Require: An aircraft assembly line scheduling problem $\mathcal{P}$ in the form of a constraint programming model (Section 4.2) and a time limit τ

Initialize solution $\mathcal{S}^* = \{(S_j^*, q_j^*, k_j^*)_{j \in V}, (r_k^*)_{k \in \mathcal{R}}\}$ by solving $\mathcal{P}$ with CP Optimizer under time limit τ'

$\pi_{ratio} \leftarrow 100$

$\tau_{base} \leftarrow 10$

$\tau_{inc} \leftarrow 0$

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

  $\mathcal{C} \leftarrow \texttt{sweep}(\mathcal{P}, \mathcal{S}^*)$ (get all the peak tasks)

  $\mathcal{C}^* \leftarrow \mathcal{C} \cup \{i \in V \mid \exists j \in \mathcal{C}, (i,j) \in E\}$ (add the tasks that precede them)

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

  $\mathcal{P}' \leftarrow \mathcal{P}$

  for $j \in V \setminus \mathcal{C}'$ **do**

    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \texttt{presenceOf}(T_{j q_j^* k_j^*})$

    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \texttt{startAt}(T_j, S_j^*)$

  end for

  $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{\sum_{k \in \mathcal{R}} r_k < \sum_{k \in \mathcal{R}} r_k^*\}$

  Get solution $\bar{\mathcal{S}} = \{(\bar{S}_j, \bar{q}_j, \bar{k}_j)_{j \in N}, (\bar{r}_k)_{k \in \mathcal{R}}\}$ by solving $\mathcal{P}'$ with CP Optimizer under time limit $\min(\tau_{base} + \tau_{inc}, \tau - \text{elapsed time})$

  if $\sum_{k \in \mathcal{R}} \bar{r}_k < \sum_{k \in \mathcal{R}} r_k^*$ **then**

    $\mathcal{S}^* \leftarrow \bar{\mathcal{S}}$

    $\tau_{inc} \leftarrow 0$

    $\pi_{ratio} \leftarrow 100$

  else if $\bar{\mathcal{S}}$ is empty **then**

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

    $\pi_{ratio} \leftarrow \max(\pi_{ratio}^{\min}, \pi_{ratio} - \nu)$

  end if

end while

return $\mathcal{S}^*$

that can be done in parallel. Within these groups, most of the tasks must be done in series. Incompatibility/disjunctive constraints and maximum 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. Given these characteristics, the computational experiments are based on the following instance sets.

Four sets of small 8-task randomly generated instances were first defined to compare the MILP formulations (see Appendix A for the results). Moreover, Sets 3 and 4 were extended in order to create instances of up to 11 tasks. Their characteristics are listed in Table 4, which gives, for each instance family, the number of tasks, the number of precedence constraints, the number of maximum time lag constraints, the number of disjunctive constraints, the total number of resources, the number of areas, and the total number of different modes.

The total number of operator profiles refers to the different types of profiles that exist in the station (e.g., electricians and pipe specialists). For all small instances, this number is set to two. As for the number of different modes, it is the sum of the modes of all tasks. Each mode is a combination of a duration, a number of operators and an operator profile. Therefore, two modes of a task can differ on the number of operators, the operator profile or both. For example, Set 1-8 has a total of 11 modes. For that instance, tasks 2 to 6 have a single mode. Tasks 0 can be done by 1 or 2 operators (always of profile 2), task 1 can be done by 2 or 3 operators of profile 2 and task 7 must be done by two operators but they can be of any of the two profiles.

Table 4: Small Instance Characteristics

Instance	$\|V\|$	$\|E\|$	$\|L\|$	$\|D\|$	$\|\mathcal{R}\|$	$\|A\|$	$\sum_{j\in V} \mathcal{M}_j$
Set1-8	8	6	1	1	2	2	11
Set2-8	8	8	1	1	2	2	20
Set3-8	8	7	1	1	2	2	17
Set4-8	8	7	1	1	2	2	15
Set3-9	9	8	1	1	2	2	18
Set3-10	10	9	1	1	2	2	19
Set3-11	11	10	1	1	2	2	20
Set4-9	9	8	1	1	2	2	16
Set4-10	10	9	1	1	2	2	17
Set4-11	11	10	1	1	2	2	18

Then, we consider two moderate-sized real instances from the Aircraft Final Assembly Line. Their characteristics are summarized in Table 5.

The small and the moderate-size real instances have been created with a similar density of constraints (precedence, maximum time lags and number of modes per task) than the real industrial instances.

Finally we consider larger industrial instances from the aircraft manufacturer assembly line, denoted by letters from A to G. They have between

Table 5: Moderate-Size Real Instance Characteristics

Instance	$\|V\|$	$\|E\|$	$\|L\|$	$\|D\|$	$\|\mathcal{R}\|$	$\|A\|$	$\sum_{j \in V} \mathcal{M}_j$
Set1-70	70	64	0	11	4	2	84
Set1-100	100	90	3	14	4	2	134

Table 6: Large-Size Real Instance Characteristics

Instance	$\|V\|$	$\|E\|$	$\|L\|$	$\|D\|$	$\|\mathcal{R}\|$	$\|A\|$	$\sum_{j \in V} \mathcal{M}_j$
A	90	58	4	2	2	3	101
B	159	138	4	3	2	2	239
C	455	364	200	53	7	1	495
D	455	364	200	75	7	1	495
E	721	1718	387	53	7	4	764
F	486	631	13	75	6	1	540
G	165	129	5	2	2	1	195

Table 7: Large-Size Real Instance Additional Characteristics

Instance	Network Complexity	Critical Path Length (CPL)	CPL - $\|T\|$	Is GSP
A	0.98	452	0	No
B	1.11	382	123	No
C	1.45	56	344	No
D	1.45	56	344	No
E	2.97	101	620	No
F	1.35	249	421	No
G	1.15	182	298	No

90 to 721 tasks each. Their characteristics are shown in Table 6. They belong to real stations of three different final assembly lines. As an example, instances C and D are from final test assembly stations. In those stations, there are more incompatibility and maximum time lag constraints. Instance E is the largest one. It is issued from the main structural joint of one of the final assembly lines and it requires not only a high number of tasks but also more operator profiles. For these instances, we have also calculated additional characteristics, which are displayed in Table 7. In this table, the first column is the network complexity, which is defined by the average number of arcs of each node in the activity graph. The second column shows the length of the critical path in the activity network, considering that the mode with the largest duration is selected for each task. The third column is the difference between the reference cycle time and the length of the critical path, illustrating the margin available for optimizing the peak resource demand. Lastly, the last column shows whether the activity network of those instances is General Series Parallel, which would suggest that these instances are somehow easier to solve. None of the instances has this property.

For the instance F, Figure 1 shows the resource profiles obtained in the software used by the company. Vertically, each slot correspond to a 24-hours interval. On the top of the figure, the used resource profiles are listed horizontally. For each profile, the blue bar gives the usage interval of each profile. Clicking on a profile (see Profile 4) provides the interval of each task using the corresponding profile. In the bottom, the number of operators per profile used along the cycle time is displayed as a stacked bar graph, each profile having a different color. It can be seen that not all the profiles have the same workload. That is a common characteristic: there are some loaded profiles and some others, very specific, with few tasks assigned.

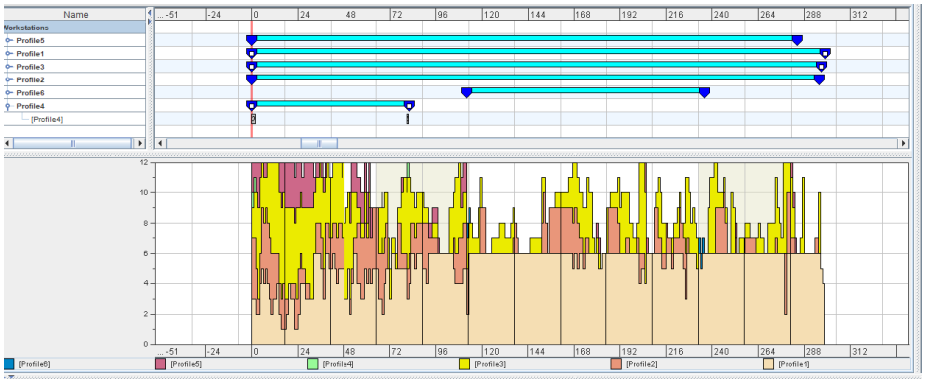

Figure 1: Resource usage for instance F

6.2 Comparison between CP and MILP Approaches

The instances used to compare the MILP and CP formulations are the small instances described in Table 4 and the moderate-sized industrial instances of Table 5. Due to the use of CP Optimizer 12.6.0 to solve the CP formulation, 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 by CP. 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 8. The results clearly establish the superiority of the CP Optimizer solver compared to the MILP models on the considered scheduling problem.

Table 8: Solution times of MILP approaches and CP Optimizer

Inst	$\|T\|$	SEE-M (STD) t(s)	OOE-M (STD) t(s)	OOE-M (MIN) t(s)	CP optimizer t(s)
Set1-8	1150	21.09	1.93	0.67	0.01
Set2-8	3475	19.66	0.58	0.47	0.02
Set3-8	1400	121.98	8.24	0.22	0.13
Set4-8	1300	36.58	3.82	0.11	0.02
Set3-11	1700	4133	452.53	1.45	0.29
Set4-11	1300	473.54	76.32	13.07	0.05
Set1-70	120000	—	—	—	2.3
Set1-100	170000	—	—	—	20.7

6.3 Comparison of LNS with CP Optimizer and the heuristic used in practice

Table 9: Objective values reached by CP Optimizer and LNS

Inst	$\|V\|$	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 10: LNS Improvement over CP Optimizer objective value

Inst	$\|V\|$	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%

In order to assess the LNS algorithm efficiency as a heuristic, we use the set of industrial instances A, B, C, D, E, F and G.

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 9 displays the objective value of the best solutions found by both algorithm within the time limits. In the first column, the number of tasks 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 method 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 10, that displays the improvement as a percentage, shows that when the time limit is 15 minutes or 30 minutes, the LNS approach never underperforms 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 schedule generation scheme (SSGS) heuristic deployed at the aircraft manufacturer, described in Section 3. This comparison is shown in Table 11. It can be seen that LNS leads to better solutions in 4 out of the 6 instances. For the two where it came to the same solution (C, D), LNS was able to prove the optimality as all tasks could be freed, resorting to the CP model. Moreover, for instance G, the heuristic is unable to find a feasible solution while LNS provided one.

6.4 Impact of cycle time on the objective function

As stated in Section 3, during the tactical planning, another frequent decision is to choose between a set of cycle times for the assembly line. In consequence, in this section we evaluate the impact of cycle time on the objective function for both CP and LNS algorithms. We run the experiments with a duration of 15 minutes on the A-G instances with a

Table 11: LNS - SSGS Heuristic Comparison

Inst	$\|V\|$	Heuristic	LNS	Improvement
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	∞

new cycle time $|T'|$, such that $|T'| = \rho|T|$ where $|T|$ is the cycle time used in the previous experiments and ρ varies in $[0.2, 2]$. The results are displayed in Table 12. First, we can see that for most instances, when $\rho \leq 0.8$ neither of the two methods can find a feasible solution, which suggests that the instances become infeasible. Then, unsurprisingly, the objective value improves as the cycle time increases, and this is particularly visible for the B and G instances, where the target value is increased by at least 50 %. If we want to compare the two algorithms, we can observe that LNS is more interesting when the cycle time value has small variations compared to the reference value, and the performance of both algorithms becomes comparable when the cycle time becomes high. This can be explained in two ways. Firstly, the problem can become very easy to solve. Secondly, when this happens the solution may have a very flat resource utilization profile, which makes the neighborhood of the LNS method extremely large and deteriorates its performance. The same can happen when the cycle time gets too "tight".

Table 12: Objective Value with varying Cycle Time

	A		B		C		D		E		F		G	
ρ	CP	LNS	CP	LNS	CP	LNS	CP	LNS	CP	LNS	CP	LNS	CP	LNS
0.2	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅
0.4	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅
0.6	∅	∅	∅	∅	21	21	23	22	∅	∅	23	26	∅	∅
0.8	∅	∅	20	13	21	19	22	19	24	21	20	18	∅	∅
1	6	6	13	9	19	19	20	18	24	21	20	17	9	9
1.2	6	6	8	8	18	19	20	18	22	21	18	17	8	8
1.4	6	6	7	7	18	20	20	20	20	19	18	17	7	8
1.6	5	5	6	6	18	19	20	20	19	19	18	17	5	5
1.8	4	4	6	6	18	18	20	19	19	19	18	18	5	5
2	4	4	6	6	18	18	20	18	19	19	18	18	4	4

6.5 Experiments on the multi-mode resource investment problem

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 13.

Table 13: LNS Improvement over CP Optimizer objective value 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%

Looking at the results, we can make several remarks. 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. Note that the improvement after 1 minute is already significant. It can be noted finally that the superiority of LNS increases with the size of the instances.

7 Conclusions & further research

In this work, we have proposed new methods for the aircraft assembly line scheduling problem. We have first proposed a new on-off event based formulation that extends the one proposed by Koné et al. [2011] to maximal time lags, multiple modes and the resource leveling objective. The computational results from Appendix A have shown that the new model is able to solve only small instances to optimality although it gets better results than the start-end event-based formulation previously proposed by Borreguero-Sanchidrián et al. [2014]. We have also shown that a standard constraint programming model solved with CP Optimizer significantly outperforms the MILP approaches. Remarking that the methods embed-

ded 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 have proposed a new LNS method more adapted to the resource leveling objective. We have 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).

As a future research direction, the proposed method could be extended to more complex variants to better tackle other industrial needs. As shown by our experiments on the resource investment problem, our model extends easily and efficiently to multiple renewable and non-renewable resources as well as to weighted resource leveling. However, it would be interesting to include the assignment of multi-skilled resources, which would solve both the tactical resource leveling and operational resource assignment in an integrated manner. This would require significant changes to the constraint programming model, using for instance the model proposed by Young et al. [2017]. Ergonomic constraints and objectives for operators welfare could also be considered for the problem at hand, as recently proposed in a related problem by Arkhipov et al. [2018].

From the methodological point of view, additional filtering techniques could be developed to solve the problem under study. The CP solution scheme encompasses also constraint propagation techniques that can be used as preprocessing 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].

Acknowledgments

This work has been partially funded by the French National Research Agency (ANR) project ANR-18-CE10-0007 “PER4MANCE” and the Artificial and Natural Intelligence Toulouse Institute (ANITI).

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

R. Alvarez-Valdés and J.M. Tamarit. The project scheduling polyhedron: Dimension, facets and lifting theorems. *European Journal of Opera-*

tional Research, 67:204–220, 1993.

Dmitry Arkhipov, Olga Battaïa, Julien Cegarra, and Alexander Lazarev. Operator assignment problem in aircraft assembly lines: A new planning approach taking into account economic and ergonomic constraints. In *7th CIRP Conference on Assembly Technologies and Systems (CATS 2018)*, volume 76, pages 63–66, 2018.

C. Artigues and E. Hébrard. 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)*, pages 814–819, Ghent, Belgium, 2013.

C. Artigues, P. Michelon, and S. Reusser. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249–267, 2003. ISSN 0377-2217.

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

Christian Artigues, Oumar Koné, Pierre Lopez, and Marcel Mongeau. *Mixed-Integer Linear Programming Formulations*, pages 17–41. Springer International Publishing, Cham, 2015. ISBN 978-3-319-05443-8.

Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. International Series in Operations Research and Management Science. Kluwer, 2001.

Alexander Biele and Lars Mönch. Hybrid approaches to optimize mixed-model assembly lines in low-volume manufacturing. *Journal of Heuristics*, 24(1):49–81, 2018.

T. Borreguero, A. García Sánchez, and M. Ortega Mier. 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, 2015a.

T Borreguero, F Mas, JL Menéndez, and MA Barreda. Enhanced assembly line balancing and scheduling methodology for the aeronautical industry. *Procedia engineering*, 132:990–997, 2015b.

Tamara Borreguero. *Scheduling with limited resources along the aeronautical supply chain: From parts manufacturing plants to final assembly lines*. PhD thesis, Universidad Politécnica de Madrid, 2020.

Tamara Borreguero, Christian Artigues, Álvaro García Sánchez, Miguel Ortega Mier, and Pierre Lopez. Multimode time-constrained scheduling problems with generalized temporal constraints and labor skills. In *7th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*, pages 809–813, Prague, Czech Republic, 2015c.

Tamara Borreguero-Sanchidrián, Álvaro García-Sánchez, and Miguel Ortega Mier. A MILP event based formulation for a real-world multimode RCSP with generalized temporal constraints. In *Managing Complexity*, pages 113–120. Springer, 2014.

Marco Bortolini, Francesca Calabrese, Francesco Gabriele Galizia, Cristina Mora, and Valentina Ventura. Industry 4.0 technologies: A cross-sector industry-based analysis. In *Proceedings of the International Conference on Sustainable Design and Manufacturing*, pages 140–148. Springer, 2021.

Nils Boysen, Philipp Schulze, and Armin Scholl. Assembly line balancing: What happened in the last fifteen years? *European Journal of Operational Research*, 301(3):797–814, 2022.

J Brimberg, WJ Hurley, and RE Wright. Scheduling workers in a constricted area. *Naval Research Logistics (NRL)*, 43(1):143–149, 1996.

P. Brucker and S. Knust. *Complex Scheduling*. GOR-Publications. Springer, 2011. ISBN 9783642239281.

Jens Buergin, Sina Helming, Jan Andreas, Philippe Blaettchen, Yannick Schweizer, Frank Bitte, Benjamin Haefner, and Gisela Lanza. Local order scheduling for mixed-model assembly lines in the aircraft manufacturing industry. *Production Engineering*, 12:759–767, 2018.

José Coelho and Mario Vanhoucke. Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research*, 213(1):73–82, 2011.

Jean-François Cordeau, Gilbert Laporte, Federico Pasin, and Stefan Ropke. Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409, 2010.

Eamonn T Coughlan, Marco E Lübbecke, and Jens Schulz. A branch-price-and-cut algorithm for multi-mode resource leveling. *European Journal of Operational Research*, 245(1):70–80, 2015.

Bert De Reyck and Willy Herroelen. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119(2):538–556, 1999.

Erik Demeulemeester. Minimizing resource availability costs in time-limited project networks. *Management Science*, 41(10):1590–1598, 1995.

Erik Demeulemeester and Willy Herroelen. *Project Scheduling: A Research Handbook*. International Series in Operations Research & Management Science. Springer, 01 2002.

Andreas Drexl and Juergen Gruenewald. Non-preemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 25(5):74–81, 1993.

Patrick Gerhards. The multi-mode resource investment problem: A benchmark library and a computational study of lower and upper bounds. *OR Spectrum*, 42(4):901–933, 2020.

Daniel Godard, Philippe Laborie, and Wim Nuijten. Randomized large neighborhood search for cumulative scheduling. In *ICAPS*, volume 5, pages 81–89, 2005.

S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010. ISSN 0377-2217.

Vera C. Hemmelmayr, Jean-François Cordeau, and Teodor Gabriel Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228, 2012.

H. Kagermann and W. Wahlster. 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*, 2013.

Rainer Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333, 1996.

Rainer Kolisch. *Make-to-order assembly management*. Springer, 2000.

Oumar Koné, Christian Artigues, Pierre Lopez, and Marcel Mongeau. Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, 38(1):3–13, 2011. ISSN 0305-0548.

Stefan Kreter, Andreas Schutt, and Peter J Stuckey. Using constraint programming for solving RCPSP/max-cal. *Constraints*, 22(3):432–462, 2017.

Philippe Laborie and Daniel Godard. Self-adapting large neighborhood search: Application to single-mode scheduling problems. volume 8, Paris, 2007.

Philippe Laborie, Jérôme Rogerie, Paul Shaw, and Petr Vilím. IBM ILOG CP Optimizer for scheduling. *Constraints*, 23(2):210–250, 2018.

Broos Maenhout and Mario Vanhoucke. An integrated nurse staffing and scheduling analysis for longer-term nursing staff allocation problems. *Omega*, 41(2):485–499, 2013.

Fernando Mas, José Luis Menéndez, Manuel Oliva, Javier Servan, Rebeca Arista, and Carmelo Del Valle. Design within complex environments: Collaborative engineering in the aerospace industry. In *Information System Development*, pages 197–205. Springer, 2014.

Fernando Mas, Rebeca Arista, Manuel Oliva, Bruce Hiebert, Ian Gilkerson, and José Ríos. A review of PLM impact on US and EU aerospace industry. *Procedia Engineering*, 132:1053–1060, 2015.

Fernando Mas, José Ríos, Alejandro Gómez, and Juan Carlos Hernández. Knowledge-based application to define aircraft final assembly lines at the industrialisation conceptual design phase. *International Journal of Computer Integrated Manufacturing*, 29(6):677–691, 2016.

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

Mireille Palpant, Christian Artigues, and Philippe Michelon. LSSPER: Solving the resource-constrained project scheduling problem with large neighbourhood search. *Annals of Operations Research*, 131(1-4):237–257, 2004.

Oliver Polo Mejía, Christian Artigues, Pierre Lopez, and Virginie Basini. Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility. *International Journal of Production Research*, 58(23):7149–7166, 2020.

A. Alan B. Pritsker, Lawrence J. Watters, and Philip M. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108, 1969. ISSN 00251909.

Christoph Schwindt and J. Zimmermann. *Handbook on Project Management and Scheduling Vol.1*. Springer, 01 2015.

Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael J. Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming - CP'98, 4th International Conference*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431, Pisa, Italy, 1998. Springer.

Charles Thomas and Pierre Schaus. Revisiting the self-adaptive large neighborhood search. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 557–566. Springer, 2018.

Hong Wang, Tongling Li, and Dan Lin. Efficient genetic algorithm for resource-constrained project scheduling problem. *Transactions of Tianjin University*, 16(5):376–382, 2010. ISSN 1006-4982.

Jan Weglarz, Joanna Józefowska, Marek Mika, and Grzegorz Waligóra. Project scheduling with finite or infinite number of activity processing modes – A survey. *European Journal of Operational Research*, 208(3): 177–205, 2011. ISSN 0377-2217.

Kenneth D Young, Thibaut Feydy, and Andreas Schutt. Constraint programming applied to the multi-skill project scheduling problem. In *Principles and Practice of Constraint Programming: 23rd International Conference (CP 2017)*, pages 308–317, Melbourne, VIC, Australia, 2017. Springer.

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

A Experimental comparison of MILP formulations

In this appendix, the MILP models were validated and compared on the small randomly generated instances described in Table 4 of Section 6.1 The computational results were obtained using CPLEX 12.4 solver. The tests were carried out on an Intel Core i7 2630QM processor with 2 GHz and 4 GB RAM, running Windows 7.

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 the SEE-M formulation proposed in Borreguero-Sanchidrián et al. [2014], the instances took times from seconds to fifteen minutes, and from 0.1 second to eight minutes for the new OOE-M formulation. The harder instance to solve was Set3-11, that took up to 4133 seconds for a time horizon of 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 14 shows an example of this evolution for each of the formulations.

Table 14: Sample Solving Time for Different Cycle Times

	Instance	$\|T\|$ 31.5	$\|T\|$ 33	$\|T\|$ 34.75	$\|T\|$ 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 model 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, the number of events was reduced by one unit at a time. Then, the model was solved and we checked that the solution was still optimal. The solution time increases exponentially with the number of events, even when solving the same instance.

Figure 2 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 $T = 11.5$ days. For example, for the SEE-M formulation the first set (Set1-8), when solved for $T = 11.5$ 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 OOE-M model.

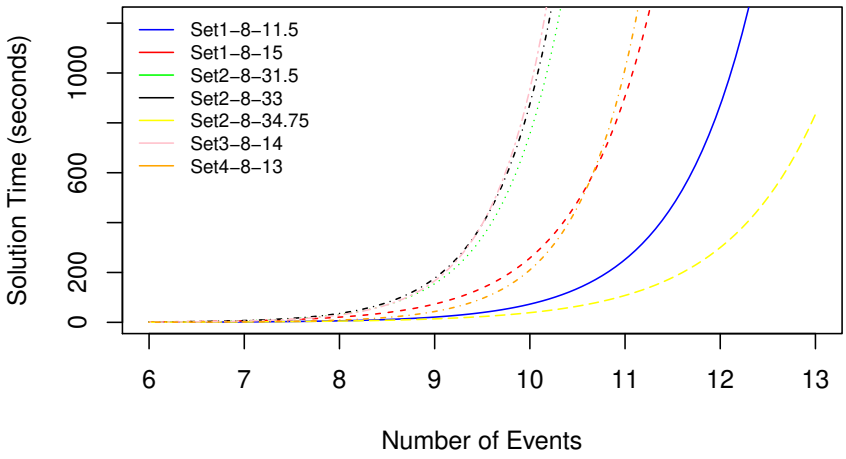

Figure 2: SEE-M solution time *vs.* number of events

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. Nevertheless, reducing too much the number of events could lead to infeasible or non-optimal solutions. Therefore, some additional research on the instance characteristics (network complexity, resource factor) that can help to predict the minimum number of events that leads to optimal solutions would be needed.

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 3 shows the histogram for the time spent for getting the optimal solution with the OOE-M formulation by the time spent with 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.

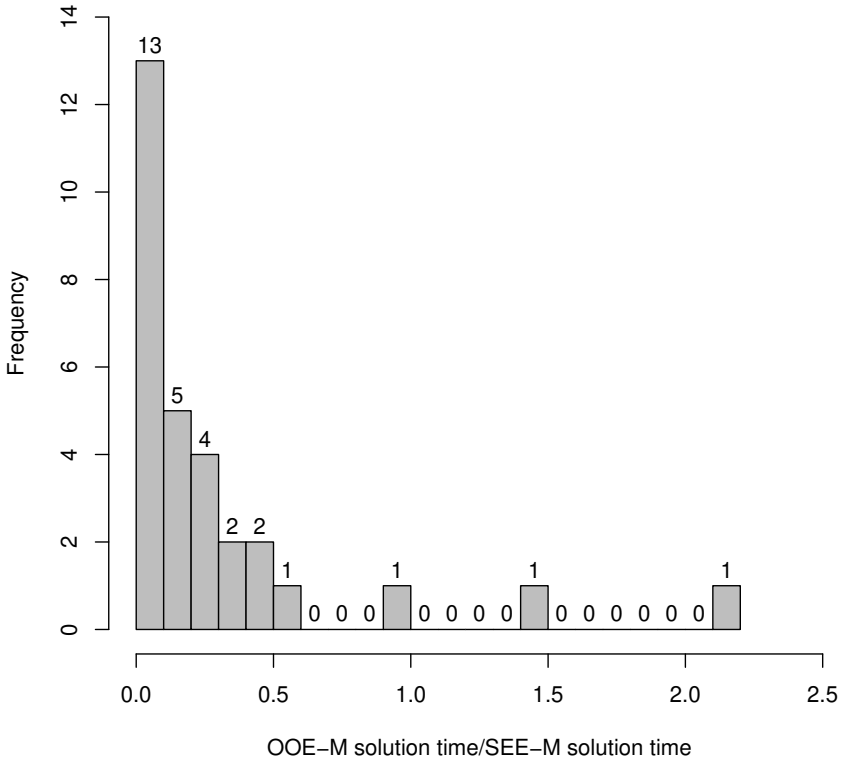

Figure 3: OOE-M Solution Time / SEE-M Solution Time

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

instances.