Assembly Line Preliminary Design Optimization for an Aircraft

Stéphanie Roussel[1]
ONERA, ONERA DTIS, Toulouse, Université de Toulouse, France

Thomas Polacsek
ONERA, ONERA DTIS, Toulouse, Université de Toulouse, France

Anouck Chan
ONERA, ONERA DTIS, Toulouse, Université de Toulouse, France

Abstract

In the aeronautics industry, each aircraft family has a dedicated manufacturing system. This system is classically designed once the aircraft design is completely finished, which might lead to poor performance. To mitigate this issue, a strategy is to take into account the production system as early as possible in the aircraft design process. In this work, we define the Assembly Line Preliminary Design Problem, which consists in defining, for a given aircraft design, the best assembly line layout and the type and number of machines equipping each workstation. We propose a Constraint Programming encoding for that problem, along with an algorithm based on epsilon constraint for exploring the set of Pareto solutions. We present experiments run on a set of real industrial data. The results show that the approach is promising and offers support to experts in order to compare aircraft designs with each other.

2012 ACM Subject Classification Applied computing → Computer-aided manufacturing

Keywords and phrases Assembly line design, Constraint Programming, Multi-objective, Industry 4.0

Digital Object Identifier 10.4230/LIPIcs.CP.2023.32

Supplementary Material *Dataset (Assembly Line Problems)*: `https://github.com/stephroussel/assemblyLine`

1 Introduction

The aeronautics industry is highly specialized. It requires significant investments in research to design and manufacture a new aircraft. Each new aircraft family, consisting of several models within a given size range, requires the development of a dedicated industrial system. The latter includes the manufacturing and assembly processes, supply chain, tooling and facilities necessary to produce an aircraft at scale. The design of such a system typically involves several steps of high-level objectives refinements, which can take several years and requires collaboration between various stakeholders (aircraft manufacturers, suppliers, and regulatory agencies, etc.). Once established, the system can be used for the entire aircraft family lifecycle, which can last several decades. While aircraft manufacturers can generally leverage existing resources such as buildings and infrastructure, they have to develop new manufacturing and assembly lines to meet the specific requirements of each aircraft family.

An industrial system has its own set of requirements and is designed so as to optimize several criteria. For instance, assembly lines must comply to organizational constraints and ergonomics recommendations while minimizing the overall investment cost. As the manufacturing system depends highly on the aircraft choices (*e.g.* the aircraft material such as carbon fibers, metal, etc.), the manufacturing system design is generally thought after the

[1] Corresponding author

© Stéphanie Roussel, Thomas Polacsek, and Anouck Chan;
licensed under Creative Commons License CC-BY 4.0

aircraft design. This might result in a low performance of the latter or even the impossibility to meet some requirements. For instance, choosing carbon fibers on the aircraft side requires buying autoclaves on the industrial system side, with an associated cost that exceeds a given investment budget.

Therefore, a nowadays trend in aeronautics is to take into account the industrial system design as early as possible in the aircraft design process ([18, 22]). A way to do so, and the approach we follow in this paper, is to design the best preliminary industrial system for an aircraft design that is not validated yet and assess its performance. This allows to estimate whether expected industrial objectives can be met for a given aircraft. Moreover, if several aircraft designs are candidates, this also allows comparison between them with respect to their industrial suitability. Finally, such an approach offers support for making trade-offs between the aircraft and the manufacturing. More precisely, a modification of the aircraft that downgrades its performance might allow to decrease significantly the cost associated to the production system. For instance, a change in the aircraft shape might increase a little bit its fuel consumption but reduce significantly the time required to build it because the new shape allows to use robots for making junctions.

This work presents contributions produced in the context of an industrial project with an aircraft manufacturer for supporting trade-offs between an aircraft and its industrial system. The objective of this project was to focus on one element of the industrial system, namely the *assembly line*. An aircraft assembly line is responsible for bringing together all of the components of the aircraft, including the fuselage, wings, engines and interiors, into a complete and fully functional aircraft. The assembly lines we consider are pulsed and composed of several workstations that are equipped with specific machines. Assembly lines must be designed to be flexible enough to accommodate changes in the manufacturing process as the aircraft family evolves over time. Moreover, because of the international context, production rates can vary a lot within the aircraft lifecycle. It is therefore essential to choose an aircraft design and an assembly line design that are compatible with such evolution. Other elements of the industrial system, *e.g.* the supply chain, depend also highly from the aircraft design but they were out of scope of our project. Nevertheless, the assembly line by itself represents a quite expensive part of the industrial system. In fact, when an aircraft design allows to have one workstation (or one machine) less than another one, it can correspond to costs of several million euros.

In the preliminary stages of designing an assembly line, aircraft manufacturers can use the PERT (Program Evaluation and Review Technique). PERT is a network diagram that shows the sequence of tasks and their dependencies. It allows to identify critical paths and potential bottlenecks. However, PERT is a very optimistic view of the building process. In fact, because it does not take resources into account, it assumes that all tasks that are not dependent can be performed in parallel. In order to have a more realistic assessment on the assembly line, industrial architects design workstations and equip them, assign tasks to workstations and then compute a fine-grain line balancing that takes into account workers and their profiles. However, such an iterative process can be quite heavy to put in place for each aircraft design candidate.

In this paper, we present an aid-decision tool that we have developed in the context of the previously mentioned project. For a given aircraft, the tool returns the best preliminary assembly line designs with respect to several criteria. These criteria deal with the assembly line layout (workstations and associated equipment), its cost and also with the production duration. We call this problem the Assembly Line Preliminary Design Problem, which can be seen as a Resource-Constrained Project Scheduling Problem (RCPSP) with additional

constraints ([9]). The tool relies on a Constraint Programming encoding of the problem. In fact, Constraint Programming is particularly suited for handling resource and temporal constraints inherent in the assembly line design problem ([16]). In order to explore the Pareto solutions set, we define an epsilon constraint based algorithm ([4]). It basically consists in computing bounds for each criteria, choosing an exploration order and a number of points in the front, and next running a mono-criterion algorithm to generate Pareto solutions.

Our contribution can be summarized as follows:

- we present a new problem, namely the Assembly Line Preliminary Design Problem, and formalize associated constraints and criteria;
- we define a Constraint Programming encoding for this problem;
- we develop an epsilon constraint based algorithmic approach for exploring the Pareto front;
- we present experiments on a set of real data coming from an aircraft manufacturer;
- we make those data available for the community.

The paper is structured as follows. In Section 2, we describe related works. Section 3 is dedicated to an informal description of the problem and some prior work for eliciting constraints and criteria in the case of a preliminary design. In Section 4, we formally model the Assembly Line Preliminary Design Problem and propose an associated Constraint Programming encoding. We detail the algorithmic approach for exploring the Pareto solutions set in Section 5. In Section 6, we describe the real world benchmarks we have used and discuss the results obtained with the paper's approach. Finally, we conclude on future works in Section 7.

2 Related Works

A vast body of literature exists on the production and line balancing problems research topic ([5, 8]). The *Simple Assembly Line Balancing Problem* (SALBP) consists in assigning tasks to workstations on a single straight line with respect to some precedence constraint and to a fixed workstation capacity. Such a problem can be seen as a bin packing problem with additional constraints due to precedence. Depending on the objective function, there are two classical extensions: *SALBP-1* in which the number of workstations is minimized for a given production rate and *SALBP-2* in which the production rate is maximized for a given number of stations.

Assembly Line Design and Constraint Programming. When considering the design of assembly lines, additional features are generally taken into account, in particular resources required for the task execution ([24]). Resources are generally used to model machines or equipment that have to be positioned on the assembly line but also to model workers performing assembly operations. Constraint Programming (CP) models have been developed for solving various assembly line problems, and specifically problems in which resources are involved. In [11], the authors present encodings for solving SALBP-1, SALBP-2 and the Task Assignment and Equipment Selection Problem (TAESP). In the latter, one type of equipment has to be assigned to each station and the objective is to minimize the total associated cost. In [2], the authors use CP for solving the Resource-Constrained Assembly Line Balancing Problem (RCALBP) in which there can be several resources candidates for performing each task. The objective is the cycle time minimization, *i.e.* the time spent in each workstation of the line. This work has been extended in [1] into the Generalized RCALBP, in which several

modes with different resources consumption are considered for each task. These modes are expressed through conjunction normal form formulas that are handled directly by the CP solver through the use of *AND* and *OR* operators.

Multi-criteria Line Design. As described in [24], the assembly design problem often comes to the optimization of several criteria. In [29], the authors consider the minimization of the number of workstations, the minimization of cycle time and maximization of workload smoothness and of work relatedness. Each criterion is studied independently, except for the last two. The authors of [13] consider the problem of equipping workstations with respect to some exclusion and compatibility constraints between pieces of equipment. They study several criteria: minimization of the investment cost, maximization of the line throughput rate, minimize the occupied space and minimize the workstations complexity. In [19], the assembly line considered is mixed-model, meaning that similar models of a product are produced on the same line. Their objective is to minimize the idle time of each model on the line and minimize the total equipment cost. More recently, in [20], the authors optimize the idle time and the unit product costs. All those works mainly use evolutionary algorithms for computing a set of non-dominated solutions. An extensive comparison between 34 algorithms is provided in [20]. In [15], the authors address the two-sided assembly line balancing problem with multi-operator workstations with respect to several criteria: minimization of the number of mated workstation, minimization of the number of workstation and minimization of the number of workers. Those criteria are optimized sequentially. A CP and an ILP (Integer Linear Programming) approaches, that both consider the time at which tasks start and end, are proposed.

Line Balancing in the Aeronautical Industry. In the context of assembly line balancing for the aeronautical industry, literature is less extensive. The authors of [14] consider the efficiency of labor utilization, specifically for tasks that consist in preparing the aircraft for assembly, for instance when it arrives on a workstation. In a generic context of producing in low-volume, which applies for the aircraft industry, the article [6] assigns workers to station and start times to each task so as to minimize the costs of total labor and inventory costs. The authors of [10] consider the local order scheduling for mixed-model assembly lines. In [7] and more recently in [26], exact (Mixed Integer Linear Programming and CP) and heuristic methods are studied for solving a mixed-model assembly line problem in which tasks have been assigned to workstations and the objective is to minimize the overall number of operators. In [3], the authors compare CP and ILP approaches for assigning workers to tasks so as to minimize the overall makespan and while ensuring that some ergonomics constraints are satisfied. Both approaches consider not only the assignment of worker to station but also the precise scheduling of all activities. In [23], a scheduling tool is developed for bridging the gap between aircraft design and aircraft manufacturing. Such a tool relies on a local search approach and is compared with a CP encoding. The problem consists in minimizing the makespan. As done in the present paper, it considers aircraft zones that limit the number of workers that can work simultaneously. However, it does not take into account machines on the assembly line or zone neutralization and does not allow to optimize the takt-time.

Finally, [12] is a previous work in which we present results we have obtained on the same industrial project. In that work, we focus on the elicitation part and present a Goal-Oriented Requirements Engineering methodology that allowed us to obtain constraints and criteria associated with an assembly line performance. We indicate that an operations research approach was used to explore the set of assembly line solutions but this approach is not formally described and only returns a few assembly line designs solutions.

RCPSP and Multi-Objective Optimization. As explained in [26], the assembly line balancing problems in the aeronautical industry can be modeled into RCPSPs with some additional constraints, such as temporal constraints or incompatibility constraints. Constraint Programming provides a expressive language for such constraints, along with efficient solvers [16].

Some works of the literature address the Multi-Objective Constrained Optimization Problems (MO-COP). In [25], the authors propose a multi-objective lower bound set that allows to detect inconsistency of a multi-objective problem. The authors of [21] define an interactive algorithm for MO-COP, that gradually refines a set of Pareto solutions by exploring regions chosen by the user in which a Pareto front might exist. The specific case of Multi-Objective in RCPSP has also received attention, as shown in [4].

3 Preliminary Assembly Line Design Description

In this section, we describe the assembly lines we have considered, namely pulse assembly lines. Then, we briefly present the work that we have done prior to optimization for eliciting constraints and criteria relevant for a preliminary assembly line design stage.

3.1 Pulse Assembly Line

The assembly line we consider in this work is a pulse line composed of several workstations. In aeronautics, pulse assembly lines are common solutions when it comes to high production rates.

The layout of pulse assembly lines we consider is a basic straight line. Each workstation in the line is responsible for performing a specific set of tasks in the assembly process, and the product being assembled moves from one workstation to the next until it is complete. When an aircraft enters the line, it goes to the first workstation. After a duration called *takt-time*[2], the aircraft goes to the second workstation and a new aircraft enters the first workstation. The aircraft that was on the last workstation is done and leaves the assembly line. Tasks cannot be performed when the aircraft is moving. This implies that all tasks must be assigned to one unique workstation.

The total time that the aircraft remains on the assembly line is called *leadtime*. The leadtime is equal to the multiplication of the takt-time by the number of workstations. Note that the range of values for the takt-time of a given assembly line can be quite large. In fact, when a new line is opened, the production can start with a few aircraft per month, which corresponds to large takt-times. Depending on business considerations, the production rate can reach several dozen aircraft per month (takt-time equal to a few hours).

The set of operations performed on each workstation is always the same and they are performed in the same order for all aircraft instances. Moreover, each workstation is equipped with a set of jigs and tools, or more generally machines, that are dedicated to it. Such machines can be heavy installations, such as scaffolds for supporting the aircraft, dedicated robots or some smaller tools, such as dedicated drilling devices.

Assembly lines we consider in this paper are single-model lines as there is one product to manufacture. In practice, there can be small variants between all the aircraft present on an assembly line but such differences are not taken into account in the preliminary stage design.

[2] Takt-time is generally called cycle time in the literature. However, cycle time sometimes refer to another duration in the aircraft industry.

3.2 Constraints and Criteria Elicitation

Prior to the tool implementation, a major part of the project we are involved in has been dedicated to the elicitation of preliminary assembly line performance constraints and criteria. To do so, we have followed a goal-oriented requirements engineering based that is described in details in [12].

As expected, one of the criteria for the assembly line design is to minimize the associated investment cost. The costs of designing an aircraft assembly line can vary significantly depending on a variety of factors, including the size and complexity of the aircraft family, the level of automation and technology required, and the specific requirements of the production process. In the case of the preliminary assembly line design, we focus on two types of cost.

Minimize the number of workstations. We first consider the workstations cost. In fact, each workstation occupies a floor area in factories and has therefore an associated cost. In reality, the latter depends on the size of machines equipping the workstation and on the volume of the aircraft parts manipulated. In the assembly lines we consider in this work, it is reasonable to consider that the parts manipulated are comparable and therefore that the workstations cost is uniform. Therefore, we try to minimize the number of workstations in the assembly line.

Minimize the number of machines. The second type of cost is the machines and tools cost. In fact, to build an assembly line, specialized tooling and equipment must be designed and installed. This may include robots, fixtures and specialized machinery, which can be very costly. As we are in a preliminary design stage, we only consider the sizing machines and tools of the line. Cost range of such machines can be quite large as it depends on the machine features. In this paper, we focus only on highly expensive machines. As each machine is really specific, there is almost one criterion for each type of machine. In this project, we did not have any cost information nor ranking between the machines. In fact, costs can be quite complex to compute (because it implies not only buying the machine but also maintaining it). We have therefore decided to simplify the machines cost and consider only the total number of machines minimization. However, the number of machines of each type should be carefully examined when looking into details solutions in the front.

Machines assignment and incompatibility. Machines and tools we consider can be assigned to exactly one workstation. As they can be voluminous, there exists incompatibility between some machines types. Therefore, we have an exclusion relationship between machine.

Minimize the leadtime. Leadtime reduction is a critical goal for any manufacturing process. In fact, manufacturers can reduce the amount of cash that is tied up in work-in-progress inventory, which can improve cash flow and financial performance.

Minimize the takt-time. As expected, the higher the production rate, the more efficient the assembly line. This corresponds to the takt-time minimization. On that point, we can note that a target takt-time is generally fixed when designing a new aircraft. Nevertheless, it is essential to assess the best takt-time that is compatible with a given aircraft design, in order to prepare future production rate increase.

Aircraft zones capacity. The space available for workers inside the aircraft is a major constraint. Workers may need to operate in tight spaces or confined areas to access parts of the aircraft being assembled, which can increase the errors or accidents risk. This is particularly the case for tasks that require precision and accuracy, such as drilling, fastening or installing electrical components. To model that, we consider that the aircraft is divided into zones, and that each zone has a capacity representing the maximum number of workers that can perform assembly tasks simultaneously.

Zone neutralization. Because of safety constraints or some assembly task nature, some assembly task can neutralize aircraft zones, meaning that it prevents any other task to be performed in those neutralized zones simultaneously. For instance, if a task requires the removal of a temporary floor during its execution, it prevents access to the associated zone.

Task features.

- There is one alternative for each task. Each alternative is characterized by types of machine used, zones occupied and neutralized.
- Duration of tasks is fixed.
- It is not possible to interrupt tasks during their execution (no preemption). However, as we are in preliminary design, tasks we consider are in fact macro ones. Therefore, it is possible for tasks that do not consume any machine to overlap on several workstations.

4 Assembly Line Preliminary Design Problem Definition

In this section, we formally define the Assembly Line Preliminary Design Problem (ALPDP) and present a Constraint Programming encoding for it.

4.1 Problem Definition

Assembly Line Preliminary Design Problem. An Assembly Line Preliminary Design Problem (ALPDP) is formally defined by a tuple $\langle \mathcal{Z}, \mathcal{S}, \mathcal{E}, \mathcal{T}, \mathcal{P} \rangle$ where:

- $\mathcal{Z}$ is the set of zones of the aircraft in which workers perform assembly activities. Each zone $z \in \mathcal{Z}$ has a capacity, denoted $capa_z$, that represents the number of workers that can work simultaneously in zone z;
- $\mathcal{S}$ is the set of machine skills that are required by assembly activities. Each skill $s \in \mathcal{S}$ represents a type of jig and tool (*i.e.* a type of machine) used for the aircraft assembly;
- $\mathcal{E}$ is a symmetric relation included in $\mathcal{S}^2$ that represents skills of machines that are incompatible and cannot belong to the same workstation. Note that for a given skill $s \in \mathcal{S}$, it is possible to have $(s, s) \in \mathcal{E}$: this indicates that two machines with the same skill s cannot be assigned to the same workstation;
- $\mathcal{T}$ is the set of assembly activities that must be performed. $\mathcal{T}$ can be partitioned into two subsets: a set of atomic activities denoted $\mathcal{A}$ and a set of composite activities denoted $\mathcal{C}$. Intuitively, composite activities are a group of atomic activities that allow to write precedence between activities in a compact way. They do not have a fixed duration and do not consume any resource (zone or machine);
- for each composite activity $c \in \mathcal{C}$, $\mathcal{A}_c$ is a subset of $\mathcal{A}$ that represent all the atomic tasks that are part of composite task c.
  Assumption. An atomic task is the child of at most one composite activity. Formally, for c_1 and c_2 in $\mathcal{C}^2$ such that $c_1 \neq c_2$, we have $\mathcal{A}_{c_1} \cap \mathcal{A}_{c_2} = \emptyset$;
- for each atomic activity $a \in \mathcal{A}$, we consider the following features:

- $dur_a \in \mathbb{N}^+$ is the duration of a;
- $\mathcal{S}_a \subseteq \mathcal{S}$ is the set of skills required by a. Note that, for a given skill s, an activity requires at most one machine with that skill;
- for each zone $z \in \mathcal{Z}$, $occ_{a,z} \in \mathbb{N}^+$ is the number of places in z required by activity a;
- $\mathcal{Z}_a^{neutr} \subseteq \mathcal{Z}$ is the set of zones that are neutralized by a. If zone z' belongs to $\mathcal{Z}_a^{neutr}$, then for all activity a' that occupies z' (*i.e.* such that $occ_{a',z'} > 0$), a and a' cannot temporally overlap.
  Assumption. For a given activity, the set of zones it occupies and the set of zones it neutralizes have an empty intersection;

- $\mathcal{P}$ is a relation in $\mathcal{T}^2$ that represents precedence requirements between activities. More precisely, $(t, t') \in \mathcal{P}$ if t must be finished before t' can start.
  Assumption 1. There does not exist precedence between composite activities and their children, *i.e.* for all $c \in \mathcal{C}$ and $a \in \mathcal{A}_c$, $(a, c) \notin \mathcal{P}$ and $(c, a) \notin \mathcal{P}$.
  Assumption 2. The directed graph induced by relation $\mathcal{P}$ is acyclic. To formally build such a graph, the first step is to create a node n_a for each atomic activity a and two nodes s_c and e_c for each composite activity $c \in \mathcal{C}$ that respectively represent the start and end of c. Then, arcs are added as follows. For each precedence $(a_1, a_2) \in \mathcal{A}^2$, an arc is added between n_{a_1} and n_{a_2}. For each precedence $(c_1, c_2) \in \mathcal{C}^2$, an arc is added between e_{c_1} and s_{c_2}. For each precedence $(a, c) \in \mathcal{A} \times \mathcal{C}$ (resp. $(c, a) \in \mathcal{C} \times \mathcal{A}$), an arc is added between n_a and s_c (resp. between e_c and n_a). Note that this graph construction is similar to the one proposed in [23].

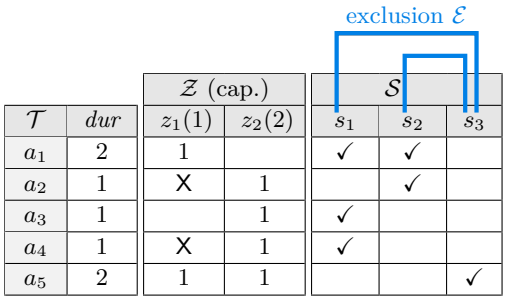
Alternative. An alternative for a ALPDP $\langle \mathcal{Z}, \mathcal{S}, \mathcal{E}, \mathcal{T}, \mathcal{P} \rangle$ is described through the following elements:

- a set of workstations $\mathcal{W}$ and for each workstation $w \in \mathcal{W}$, a start date and an end date;
- a takt-time, denoted *takt*, that represents the duration the aircraft stays on each workstation;
- for each skill s and for each workstation w, a number of machines with skill s on workstation w;
- for each activity $t \in \mathcal{T}$, a start date and an end date.

The leadtime of a solution, denoted $T_{\max}$, is the total time the aircraft spends on workstations. Formally, $T_{\max} = |\mathcal{W}| \cdot takt$.

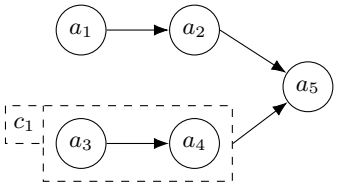
Solution. A solution for an ALPDP $\langle \mathcal{Z}, \mathcal{S}, \mathcal{E}, \mathcal{T}, \mathcal{P} \rangle$ is an alternative in which the following constraints are satisfied. Note that we only give an informal definition of constraints here, as the formal definition is detailed through the Constraint Programming encoding later in the paper.

- Start dates and end dates of workstations must represent a consistent assembly line (no temporal hole and no overlap between successive workstations) and consistent with the *takt* value (duration of a workstation is equal to *takt*);
- start dates and end dates of activities are consistent with duration (for atomic activities), with children start and end dates (for composite activities) and with precedence relationship;
- capacity of zones is never exceeded and zone neutralization is respected;
- the number of machines of each skill in each workstation allows the associated activities to be performed;
- skills exclusion is respected;
- atomic activities that consume at least one machine cannot overlap on successive workstations.



		$\mathcal{Z}$ (cap.)		$\mathcal{S}$		
$\mathcal{T}$	*dur*	$z_1(1)$	$z_2(2)$	s_1	s_2	s_3
a_1	2	1		✓	✓	
a_2	1	X	1		✓	
a_3	1		1	✓		
a_4	1	X	1	✓		
a_5	2	1	1			✓

(a) Activities description.



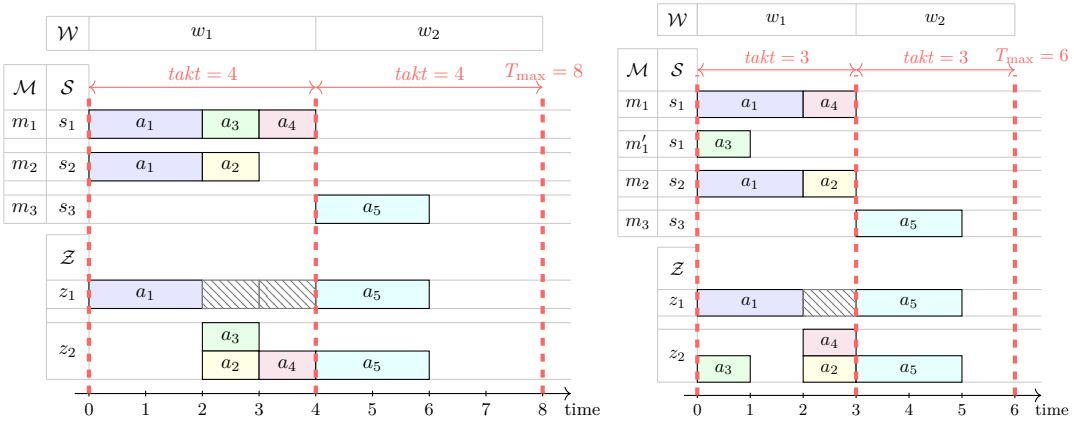
(b) Graph induced by precedence relation $\mathcal{P}$.

Figure 1 Toy example illustration.

Criteria. According to the criteria elicitation phase, we consider the following criteria: minimize the takt-time, minimize the leadtime value, minimize the total number of machines on the assembly line and minimize the number of workstations. As described earlier, we do not have any insight on how to aggregate this criteria together.

▶ **Example 1.** Figure 1 illustrates an ALPDP toy example. In this example, the set of zones $\mathcal{Z}$ is composed of two zones z_1 and z_2 that respectively have a capacity equal to 1 and 2. The set of skills $\mathcal{S}$ contains three skills, s_1, s_2 and s_3, and we consider that s_1 and s_3 exclude each other, and so do skills s_2 and s_3 (*i.e.* $\mathcal{E} = \{(s_1, s_3), (s_3, s_1), (s_2, s_3), (s_3, s_2)\}$). The set of activities $\mathcal{T}$ contains one composite activity c_1 and five atomic activities a_i with $i \in [1..5]$. Composite activity c_1 encompasses activities a_3 and a_4.
Table 1a describes atomic activities features. The X symbol indicates that an activity neutralizes a zone. The ✓symbol indicates that an activity requires a skill. For instance, the second line of the table indicates that activity a_2: has a duration equal to 1, neutralizes zone z_1, occupies one place in zone z_2, and requires skill s_2.
Precedence relation is $\mathcal{P} = \{(a_1, a_2), (a_2, a_5), (c_1, a_5), (a_3, a_4)\}$, as illustrated on Figure 1b[3].
Figure 2a illustrates a solution, denoted $sol_{3machines}$ for this ALPDP. In this solution, there are two workstations, w_1 and w_2. The first workstation contains machine m_1 with skill s_1 and machine m_2 with skill s_2. Workstation w_2 contains machine m_3 with skill s_3. Start dates of activities are respectively 0, 2, 2, 3, 4 for $a_1, \ldots, a_5$. Composite activity c_1 starts with its earliest child (here a_3 at time 2) and ends with its latest (here a_4 at time 4). In this solution, the takt is equal to 4 and the leadtime equal to 8.
A second solution, $sol_{4machines}$, is illustrated on Figure 2b. In this solution, there are two machines with skill s_1 in the first workstation. Note that for this solution, composite activity c_1 is not continuous in the sense that there is a temporal hole between its children activities. This solution allows to reach a takt equal to 3 and a leadtime equal to 6.
We can notice that $sol_{3machines}$ is better than $sol_{4machines}$ with respect to the total number of machines criteria. Both solutions are equivalent for the number of stations. However, $sol_{4machines}$ is better than $sol_{3machines}$ with respect to the takt and to the leadtime. Thus, no solution dominates the other on all criteria.

[3] Note that we do not represent explicitly start and end activities associated with c_1.



(a) Solution $sol_{3machines}$, with one machine for each skill, a takt-time equal to 4 and a leadtime equal to 8.

(b) Solution $sol_{4machines}$, with two machines for skill s_1, a takt-time equal to 4 and a leadtime equal to 8.

Figure 2 Two solution examples that each has two workstations.

4.2 Constraint Programming Encoding

We present here a Constraint Programming encoding for the ALPDP problem. For this encoding, we suppose that we are given two additional input parameters: an upper bound for the number of workstations in the assembly line, denoted nW, and an upper bound H for the leadtime.

We base our CP encoding on data structures and functions that are available in the Optimization Programming Language (OPL - [28]). In particular, we use interval variables and state functions. An interval variable encompasses a start date variable and a duration variable. It can be optional, *i.e.* it might not be present in the schedule, and it can be temporally bounded. State functions allow to express values that a function should take over given temporal intervals.

We consider the following decision variables:

- for each activity $t \in \mathcal{T}$, itv_t is an interval variable in $[0, H]$ that represented the execution of activity a. For atomic activities $a \in \mathcal{A}$, the duration of the interval is fixed to dur_a;
- for each atomic activity $a \in \mathcal{A}$, for each workstation $w \in [1..nW]$, $itv_{a,w}$ is an optional interval variable in $[0, H]$ with a duration fixed to dur_a. The interval $itv_{a,w}$ is present if and only if a starts being performed in workstation w;
- for each workstation $w \in [1..nW]$, itv_w is an optional interval variable in $[0, H]$. itv_w is present if and only if w is a used workstation, *i.e.* a workstation in which some activities are performed;
- $takt$ is an integer variable in $[1, H]$;
- for each skill $s \in \mathcal{S}$, for each workstation $w \in [1..nW]$, $skUsed_{s,w}$ is a boolean variable that indicates whether at least one machine with skill s is assigned to workstation w and $nMachines_{s,w}$ is a integer variable that represents the number of machines with skill s that are assigned to w. An upper bound for that variable is the number of atomic activities.
- for each zone $z \in \mathcal{Z}$, $stateOcc_z$ is a state function that represents the occupation state of zone z. Such a state is 1 when z is occupied by an activity being performed and is equal to 0 if z is neutralized by an activity.

With those decision variables, the ALPDP constraints can be encoded as follows.

Assembly Line Consistency Constraints.

$$\forall w \in [1..nW], \quad sizeOf(itv_w, 0) \leq takt \tag{1}$$
$$\forall w \in [1..nW], \quad sizeOf(itv_w, H) \geq takt \tag{2}$$
$$startOf(itv_1, 0) = 0 \tag{3}$$
$$\forall w \in [2..nW], \quad endAtStart(itv_{w-1}, itv_w) \tag{4}$$
$$\forall w \in [2..nW], \quad presenceOf(itv_{w-1}) \geq presenceOf(itv_w) \tag{5}$$

Constraints (1) and (2) together ensure that the duration of a used workstation is equal to the takt-time. The second parameter in the *sizeOf* function defines the value of an interval variable that is not present in the final schedule. Through Constraint (3), the first workstation starts at time 0. Constraint (4) prevents temporal holes between successive workstations. Finally, Constraint (5) ensures that no unused workstation is placed in between used workstations.

Activities and Workstation Constraints.

$$\forall c \in \mathcal{C}, \quad span(itv_c, \{itv_a | a \in \mathcal{A}_c\}) \tag{6}$$
$$\forall (t, t') \in \mathcal{P}, \quad endBeforeStart(itv_t, itv_{t'}) \tag{7}$$
$$\forall a \in \mathcal{T}, \quad alternative(itv_a, \{itv_{a,w} | w \in [1..nW]\}) \tag{8}$$
$$\forall a \in \mathcal{A}, \forall w \in [1..nW], \quad startBeforeStart(itv_w, itv_{a,w}) \tag{9}$$
$$\forall a \in \mathcal{A} \text{ s.t. } \mathcal{S}_a \neq \emptyset, \forall w \in [1..nW], \quad endBeforeEnd(itv_{a,w}, itv_w) \tag{10}$$
$$\forall a \in \mathcal{T}, \forall w \in [1..nW], \quad presenceOf(itv_{a,w}) \leq presenceOf(itv_w) \tag{11}$$
$$\forall w \in [1..nW], \quad presenceOf(itv_w) \leq \sum_{a \in \mathcal{A}} presenceOf(itv_{a,w}) \tag{12}$$
$$\forall a \in \mathcal{A}, \quad endOf(itv_a) \leq \max_{w \in [1..nW]} endOf(itv_w, 0) \tag{13}$$

Constraint (6) makes interval variables associated to composite activities span over interval variables of their children. Constraint (7) enforces precedence constraints associated with precedence relation $\mathcal{P}$. Constraint (8) uses the *alternative* constraint in CP Optimizer and guarantees that there exists exactly one workstation w such that $itv_{a,w}$ coincides with interval itv_a (only for atomic activities). Constraint (9) ensures the consistency between start dates of workstation w and of activity a if the latter starts in workstation w. Constraint (10) ensures a similar consistency for end dates but only for activities that require machines. Constraints (11) and (12) guarantees the consistency between presence of activities on workstations intervals and presence of workstation intervals. Constraint (13) ensures that workstations on which atomic activities finish are used.

Machines Constraints.

$$\forall s \in \mathcal{S}, \forall w \in [1..nW], \quad \sum_{a \in \mathcal{A}, s \in \mathcal{S}_a} pulse(itv_{a,w}, 1) \leq nMachines_{s,w} \tag{14}$$

$$\forall s \in \mathcal{S}, \forall w \in [1..nW], \quad nMachines_{s,w} \geq skUsed_{s,w} \tag{15}$$

$$\forall s \in \mathcal{S}, \forall w \in [1..nW], \quad nMachines_{s,w} \leq |\mathcal{A}| \cdot skUsed_{s,w} \tag{16}$$

$$\forall (s, s') \in \mathcal{E} \ s.t. \ s \neq s', \forall w \in [1..nW], \quad skUsed_{s,w} + skUsed_{s',w} \leq 1 \tag{17}$$

$$\forall s \in \mathcal{S} \ s.t. \ (s, s) \in \mathcal{E}, \forall w \in [1..nW], \quad nMachines_{s,w} \leq 1 \tag{18}$$

$$\forall s \in \mathcal{S} \ s.t. \ \exists a \in \mathcal{A} \ with \ s \in \mathcal{S}_a, \quad \sum_{w=1}^{nW} skUsed_{s,w} \geq 1 \tag{19}$$

$$\sum_{w=1}^{nW} \sum_{s \in \mathcal{S}} nMachines_{s,w} \geq M_{UB} \quad with \ M_{UB} = |\{s \in \mathcal{S} | \exists a \in \mathcal{A}, s \in \mathcal{S}_a\}| \tag{20}$$

Through Constraint (14), we build the consumption profile of machines with skill s by using the *pulse* primitive and we ensure that this consumption does not exceed the number of machines with skill s in the workstation. Constraints (15) and (16) express the relationship between *nMachines* and *skUsed* variables: $skUsed_{s,w}$ is equal to 0 if and only if the number of machines is equal to 0. Constraint (17) forbids to use excluded skills in the same workstation. Constraint (18) handles the specific case in which exclusion targets the same skill. Constraints (19) and (20) give an upper bound for the total number of machines.

Zones Constraints.

$$\forall z \in \mathcal{Z} \ s.t. \ capa_z = 1, \quad noOverlap(\{itv_a | occ_{a,z} > 0\}) \tag{21}$$

$$\forall z \in \mathcal{Z} \ s.t. \ capa_z > 1, \quad \sum_{\substack{a \in \mathcal{A} \\ occ_{a,z} > 0}} pulse(itv_a, occ_{a,z}) \leq capa_z \tag{22}$$

$$\forall a \in \mathcal{A}, \forall z \in \mathcal{Z} \ s.t. \ occ_{a,z} > 0, \quad alwaysEqual(stateOcc_z, itv_a, 1) \tag{23}$$

$$\forall a \in \mathcal{A}, \forall z \in \mathcal{Z}_a^{neutr}, \quad alwaysEqual(stateOcc_z, itv_a, 0) \tag{24}$$

Constraints (21) and (22) express that the capacity of zones is never exceeded (constraints (21) considers the specific case when zone have a capacity equal to 1). Finally, Constraints (23) and (24) enforce the state functions to be equal to 0 or 1 depending of occupation and neutralization of activities. As state function have one value for each instant, this prevents activities occupying a zone and activities neutralizing it to overlap.

We consider the following four criteria:

$$\textsf{minimize} \quad takt \tag{25}$$

$$\textsf{minimize} \quad \max_{w \in [1..nW]} endOf(itv_w, 0) \tag{26}$$

$$\textsf{minimize} \quad \sum_{w \in [1..nW]} presenceOf(itv_w) \tag{27}$$

$$\textsf{minimize} \quad \sum_{w \in [1..nW]} \sum_{s \in \mathcal{S}} nMachines_{s,w} \tag{28}$$

Criterion (25), (26), (27) and (28) respectively encode the minimization of the rate, the leadtime, the number of workstations in the assemby line and the total number of machines used. Note that, because of the equation $takt \cdot |\mathcal{W}| = T_{\max}$, if the takt-time and the leadtime are fixed, so is the number of workstations. This means that one of the three first criteria is redundant. Therefore, in the following, we only consider the takt-time, the leadtime and the number of machines minimization.

5 Algorithmic approach

In this section, we describe how we have used the previously defined Constraint Programming encoding to explore the Pareto solutions set.

The algorithm follows an epsilon-constraint based strategy ([4]), which requires bounds for each criteria. In our problem, there are three criteria and we therefore have to find in which order we explore them, how to fix the bounds and the number of steps on each part of the front. As the resulting procedure is not trivial, we present it in this section. We believe that this exploration algorithm could be a starting point for other problems with three or more criteria. In fact, to the best of our knowledge, the literature mostly contains epsilon-constraint based strategies for problems with two criteria.

In order to compute bounds for the criteria values, we use a lexicographic objective function. Then, for a given number of machines in the assembly line, we compute a takt-time step that allows to explore solutions. From the set of all computed solutions, we extract the Pareto ones.

The algorithm is formally described in Listing 1. In the pseudo-code, a run of the Constraint Programming encoding is represented through the function **cpSolve** that requires two parameters. The first one is the set of constraints to satisfy and the second one is the objective function to optimize. Such an objective function can be simple (one criterion) or a lexicographic order over several criteria (denoted $lex(\ldots)$). In the pseudo-code, we consider that we have a data structure for storing solution. If *sol* is a solution, then $sol.machines$, $sol.takt$ and $sol.T_{\max}$ respectively return the number of machines, the takt-time and the leadtime values associated with *sol*.

The algorithm's inputs are all constraints ((1) to (24)) of an ALPDP instance (denoted P) and a maximum number of points in the Pareto front for each number of machines. We start by computing three solutions sol^M, sol^T and sol^L that respectively minimize the number of machines, the takt and the leadtime values (lines 2-4). Solution sol^M provides a lower bound for the number of machines. The three solutions are added to the set of solutions through the function *updateFront* that maintains the set of Pareto solutions (line 5). While minimizing the takt and the leadtime, we also minimize the number of machines as the last criterion of a lexicographic objective function, which allows to compute an upper bound of the number of machines (line 6).

Next, we go through each value of the total number of machines, starting from the upper-bound. For each value, we consider the associated constraint $c_{MACHINES}$ that limits the number of machines in the assembly line (line 8). Then, we compute a lower bound and an upper bound for the takt-time, respectively through sol_1 and sol_2 (lines 9-10), that are stored in variables $takt_1$ and $takt_2$. If there is a Pareto front to explore (difference between both takt-times strictly greater than 1), we compute a step, denoted δ, for decreasing the considered Stakt-time according to the number of points in input (line 14). Starting from the takt upper-bound, we consider a constraint c_{TAKT} that fixes the takt-time and look for a solution satisfying that constraint (line 18). If it exists, the takt-value is decreased from δ (line 22) and this step is iterated until reaching the lower bound. Otherwise, there is no solution with a smaller takt value for that number of machines and the loop is stopped. The algorithm finally returns the Pareto front.

Note that the number of workstations criteria is never used in the algorithm. In fact, if two of the three criteria *takt*, *nWorkstations* and $T_{\max}$ are fixed, so is the third. As presented in the Experiments Section, results tend to show that **cpSolve** is less efficient when it comes to minimize the number of workstations. Therefore, we use the two other criteria.

Algorithm 1 Epsilon approach for ALPDP.

function $epsilonParetoALPDP(P, nMaxPointsPerMachine)$
  $sol^M \leftarrow$ **cpSolve**$(P, MACHINES)$
  $sol^T \leftarrow$ **cpSolve**$(P, lex(TAKT, LEADTIME, MACHINES))$
  $sol^L \leftarrow$ **cpSolve**$(P, lex(LEADTIME, TAKT, MACHINES))$
  $front \leftarrow updateFront(\{\}, \{sol^M, sol^R, sol^T\})$
  $m \leftarrow max(sol^L.machines, sol^T.machines)$
  while $m \geq sol^M.machines$ **do**
    $c_{MACHINES} \leftarrow$ Constraint$(\sum_{w\in[1..nW]} \sum_{s\in\mathcal{S}} nMachines_{s,w} \leq m)$
    $sol_1 \leftarrow$ **cpSolve**$(P \cup \{c_{MACHINES}\}, lex(TAKT, LEADTIME))$
    $sol_2 \leftarrow$ **cpSolve**$(P \cup \{c_{MACHINES}\}, lex(LEADTIME, TAKT))$
    $front \leftarrow updateFront(front, \{sol_1, sol_2\})$
    $takt_1 \leftarrow sol_1.takt$; $takt_2 \leftarrow sol_2.takt$
    if $takt_2 - takt_1 > 1$ **then**
      $\delta = \lfloor \frac{takt_2 - takt_1}{nMaxPointsPerMachine} \rfloor$
      $t \leftarrow takt_2 - \delta$
      $sol \leftarrow sol_1$
      while $t > takt_1 \wedge sol \neq nil$ **do**
        $c_{TAKT} \leftarrow$ Constraint$(takt = t)$
        $sol \leftarrow$ **cpSolve**$(P \cup \{c_{MACHINES}, c_{TAKT}\}, LEADTIME)$
        if $sol \neq nil$ **then**
          $front \leftarrow updateFront(front, \{sol\})$
          $t \leftarrow t - \delta$
    $m \leftarrow m - 1$
  return $front$

Functions *updateFront* and *dominates* are quite straightforward and therefore not detailed in the paper. Note that the Pareto front does not take into account the number of workstations.

6 Experiments

This section is dedicated to the experiments presentation. We first describe the benchmarks that we have used. We present two sets of experiments. First, we have solved the CP encoding with each criteria alone. This allows us to customize the **cpSolve** procedure in terms of heuristics and time out. Then, we present and analyze results associated with the Pareto solutions computation.

6.1 Benchmark Description

We have worked on a real industrial use-case dealing with the airframe assembly line of an aircraft. The airframe is the physical structure that supports all the other components of the aircraft, including avionics, passenger and cargo compartments. The materials used in its construction can include aluminium alloys, composites, titanium and other high-strength materials. The materials are cut and shaped into the various components using tools such as saws and drills. The assembly process for the airframe components typically involves careful alignment and attachment using specialised tools and techniques.

We were given three aircraft designs candidates, denoted *Design 1*, *Design 2* and *Design 3*. Features of instances are detailed in Table 1. Designs 1 and 2 have the same granularity level in terms of macro-tasks and have to be compared with each other. Design 3 is a detailed version of Design 1 with more fine-grain tasks that allows to study the scalability of the approach. These benchmarks are available at `https://github.com/stephroussel/assemblyLine`.

Table 1 Instances features: number of zones, skills, incompatible skills, tasks and atomic tasks, precedence, maximum number of workstations and time horizon.

Instance	$\|\mathcal{Z}\|$	$\|\mathcal{S}\|$	$\|\mathcal{E}\|$	$\|\mathcal{T}\|$	$\|\mathcal{A}\|$	$\|\mathcal{P}\|$	nW	H
Design 1	48	5	6	176	153	186	20	40000
Design 2	48	5	6	187	187	279	20	40000
Design 3	48	5	6	628	461	417	20	40000

In this use case, the number of skills is the same in each instance. In fact, all the designs require the same types of machines but these machines are not used the same way.

6.2 Single Objective Experiments

Experiments setup. We use IBM CP Optimizer 22.1.1 through the Java API with a timeout equal to 5 minutes. Experiments were all run on a 20-core Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz, 62GB RAM.

In order to test the CP encoding, we have first experimented it on the instances by considering one unique criterion for every run. The objective was to improve the solver setup and specifically the search strategy.

In CP Optimizer, the default search strategy is generally efficient. However, in our problem, we noticed a real performance improvement when asking the solver to instantiate first the *takt* variable. In fact, it allows to fix the duration of workstations, which is a key element for the remaining variables. Once this variable is fixed, we have tried other strategies such as deciding the presence of workstations or the number of machines in each workstation but it globally downgraded the performance compared to the solver default strategy. Therefore, we only present in this paper results associated with the *takt* variable instantiation strategy.

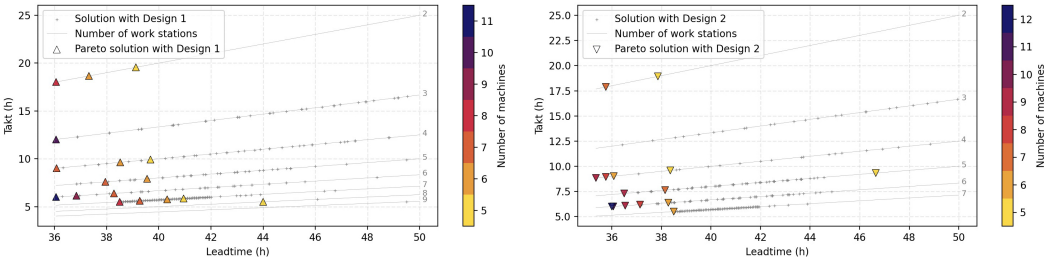
Results for the mono-objective solving are presented in Table 2 for each design and each criteria. The solver finds the optimal solution in several cases, even for the largest instance. The optimal solution is either found in less than a minute or reached the time out. Note that the solver was unable to establish the optimality of the solution with respect to the number of workstations. On that point, it would be possible to compute a lower bound using the skill exclusion relationships. We could, for instance, find the size of the largest clique in the exclusion skills graph.

Table 2 Mono-criteria CP solving with search phase customization on the *takt* variable. For each criteria, value and time (in seconds) are given. The symbol * denotes that the solution found is optimal.

Instance \ Criterion	*takt*		*nMachines*		$T_{\max}$		*nWorkstations*	
	Value	Time	Value	Time	Value	Time	Value	Time
Design 1	550*	25	5*	69	3606*	59	2	300
Design 2	550*	32	5*	28	3536	300	2	300
Design 3	340*	35	8	300	3456	300	2	300

6.3 Multi Objective Results

Experiments setup. We did not give a global timeout to the whole algorithm but instead a 1 minute timeout to each call to **cpSolve**. Note that we have let CP Optimizer handle directly the lexicographic criteria. In fact, preliminary experiments showed that there was no real performance improvement when decomposing the resolution into optimizing the first criterion and fix it, then optimizing the next one and so on.



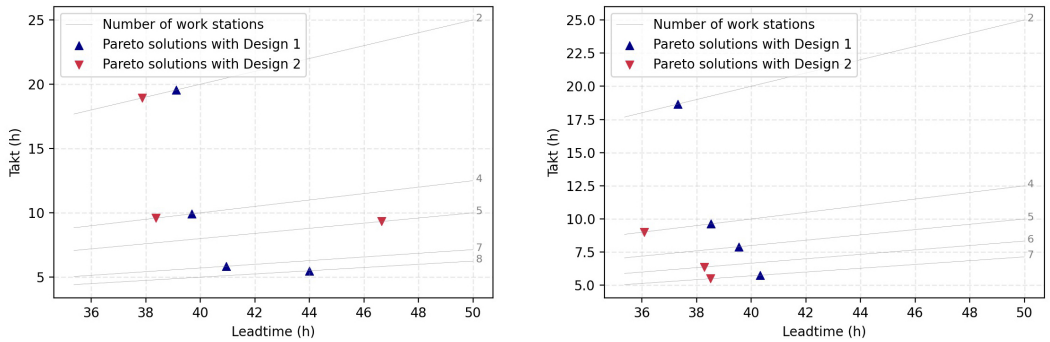
(a) Design 1 - Pareto solutions. **(b)** Design 2 - Pareto solutions.

Figure 3 Pareto solutions obtained for Designs 1 and 2.

Figures 3a and 3b show the Pareto front obtained respectively for Designs 1 and 2. The leadtime is on the horizontal axis, the takt-time on the vertical axis, the number of machines is represented through colors and the number of stations are the lines on each figure. Each small cross corresponds a to dominated solution. Note that for Design 2, we have not represented a Pareto solution that had 18 stations and a smaller takt-time.

As expected, for each design, the more machines on the assembly line, the better takt-time and leadtime values it is possible to get. Each figure shows the possible trade-offs that have to be made for each design. For instance, in Design 1, with 2 workstations, the best takt-time is equal to 18 hours. When adding 1 more workstation, such a value falls to 12 hours but requires many more machines in order to maintain the leadtime value. With one additional workstation and the same leadtime, the takt-time is less than 10 hours. We also observe that it is not worth considering more than 8 workstations for that design. Similar remarks can be made for Design 2.

Figure 4a presents Pareto solutions for Designs 1 and 2 when the assembly line is equipped with 5 machines. It shows that Design 1 (in blue on the Figure) allows to reach lower takt-time values. If one more machine is available (Figure 4b), then solutions associated with Design 2 dominate solutions of Design 1.



(a) Pareto solutions set for 5 machines. **(b)** Pareto solutions set for 6 machines.

Figure 4 Comparison of Design 1 and 2 with 5 and 6 machines.

These results have been presented to industrial partners. They have appreciated the possibility to visualize the Pareto front, which allows them to foresee the trade offs that could be made not only within the assembly line (number of workstations, number of machines, etc) but also between the aircraft design and the associated assembly line. Following results presented on Figure 4, they were surprised that Design 2 seems more promising than Design 1 only in the presence of an additional machine. Indeed, before this study, they had the false intuition than Design 2 would dominate Design 1 even with 5 machines. Such a positive feedback from end-users experts shows the added value of computing and exploring the Pareto front in this project.

7 Conclusion

In this paper, we have proposed a Constraint Programming based approach for supporting manufacturer in the early assembly design phase. We have formally defined the associated problem and have developed an algorithm for exploring the Pareto front. Each solution in this front is a trade off between the takt-time, the leadtime, the number of machines equipping the assembly line and the number of workstations. Such tools allow the aircraft manufacturer not only to assess the performance of aircraft candidates with respect to the assembly line performances but also to compare candidates with each other.

There are several directions for future works. First, the CP encoding could be improved by computing some lower and upper bounds for the targeted criteria. In fact, the addition of redundant constraints for boosting the solving might change the way the Pareto exploration should be performed and should be studied more deeply. Then, the Pareto front exploration could benefit from some recent works on computing representative Pareto solutions, such as [27]. It could also be possible to compare the CP approach and the Pareto exploration with evolutionary algorithms in terms of quality of results. We would also like to port the model to other CP solvers in order to test them. While most of the constraints could easily be written in a more classical language such as PyCSP3 ([17]), the neutralization constraints modeled by a state function in OPL might be a little trickier to encode in order to stay compact.

The instances we have used here come from a real assembly line. In order to test the approach more broadly, it would be possible to modify these instances by adding random resources or changing the precedence between tasks. Similarly, we could also adapt existing instances of the literature.

Finally, a last perspective is to consider uncertainty in the tasks duration. To do so, it might be worth considering coupling CP solvers with learning approaches that are particularly suited for managing uncertainty.

References

1 Hacı Mehmet Alakaş. General resource-constrained assembly line balancing problem: conjunction normal form based constraint programming models. *Soft Computing*, 25(8):6101–6111, 2021.

2 Hacı Mehmet Alakaş, Mehmet Pınarbaşı, and Mustafa Yüzükırmızı. Constraint programming model for resource-constrained assembly line balancing problem. *Soft Computing*, 24:5367–5375, 2020.

3 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. *Procedia CIRP*, 76:63–66, 2018.

4 Francisco Ballestín and Rosa Blanco. Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems. *Computers & Operations Research*, 38(1):51–62, 2011. Project Management and Scheduling. doi:10.1016/j.cor.2010.02.004.

5 Olga Battaïa and Alexandre Dolgui. A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, 142(2):259–277, 2013. Anticipation of risks impacts and industrial performance evaluation in distributed organizations life cycles. doi:10.1016/j.ijpe.2012.10.020.

6 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.

7 Tamara Borreguero, Alvaro García, and Miguel Ortega. Scheduling in the aeronautical industry using a mixed integer linear problem formulation. *Procedia engineering*, 132:982–989, 2015.

8 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. doi:10.1016/j.ejor.2021.11.043.

9 Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research*, 112(1):3–41, 1999.

10 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.

11 Yossi Bukchin and Tal Raviv. Constraint programming for solving various assembly line balancing problems. *Omega*, 78:57–68, 2018. doi:10.1016/j.omega.2017.06.008.

12 Anouck Chan, Anthony Fernandes Pires, Thomas Polacsek, and Stéphanie Roussel. The aircraft and its manufacturing system: From early requirements to global design. In Xavier Franch, Geert Poels, Frederik Gailly, and Monique Snoeck, editors, *Advanced Information Systems Engineering - 34th International Conference, CAiSE 2022, Leuven, Belgium, June 6-10, 2022, Proceedings*, volume 13295 of *Lecture Notes in Computer Science*, pages 164–179. Springer, 2022. doi:10.1007/978-3-031-07472-1_10.

13 Hicham Chehade, Alexandre Dolgui, Frédéric Dugardin, Lina Makdessian, and Farouk Yalaoui. Multi-objective approach for production line equipment selection. *Management and Production Engineering Review*, 3(1):4–17, 2012.

14 G. Heike, M. Ramulu, E. Sorenson, P Shanahan, and Kamran Moinzadeh. Mixed model assembly alternatives for low-volume manufacturing: the case of the aerospace industry. *International Journal of Production Economics*, 72(2):103–120, 2001.

15 Damla Kizilay and Zeynel Abidin Çil. Constraint programming approach for multi-objective two-sided assembly line balancing problem with multi-operator stations. *Engineering Optimization*, 53(8):1315–1330, 2021. doi:10.1080/0305215X.2020.1786081.

16 Philippe Laborie, Jérôme Rogerie, Paul Shaw, and Petr Vilím. IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG. *Constraints*, 23:210–250, 2018.

17 Christophe Lecoutre and Nicolas Szczepanski. PyCSP3: Modeling Combinatorial Constrained Problems in Python. Technical report, arXiv, December 2021. URL: https://hal-univ-artois.archives-ouvertes.fr/hal-03701203.

18 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: Improving Enterprise Communication*, pages 197–205. Springer, 2014.

19 Jonathan Oesterle and Lionel Amodeo. Efficient multi-objective optimization method for the mixed-model-line assembly line design problem. *Procedia CIRP*, 17:82–87, 2014. Variety Management in Manufacturing. doi:10.1016/j.procir.2014.01.038.

20 Jonathan Oesterle, Lionel Amodeo, and Farouk Yalaoui. A comparative study of multi-objective algorithms for the assembly line balancing and equipment selection problem under consideration of product design alternatives. *Journal of intelligent Manufacturing*, 30:1021–1046, 2019.

21 Tenda Okimoto, Yongjoon Joe, Atsushi Iwasaki, Toshihiro Matsui, Katsutoshi Hirayama, and Makoto Yokoo. Interactive algorithm for multi-objective constraint optimization. In Michela Milano, editor, *Principles and Practice of Constraint Programming*, pages 561–576, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

22 Thomas Polacsek, Stéphanie Roussel, Cédric Pralet, and Claude Cuiller. Design for efficient production, A model-based approach. In Manuel Kolp, Jean Vanderdonckt, Monique Snoeck, and Yves Wautelet, editors, *13th International Conference on Research Challenges in Information Science, RCIS 2019, Brussels, Belgium, May 29-31, 2019*, pages 1–6. IEEE, 2019. doi:10.1109/RCIS.2019.8877088.

23 Cédric Pralet, Stéphanie Roussel, Thomas Polacsek, François Bouissière, Claude Cuiller, Pierre-Eric Dereux, Stéphane Kersuzan, and Marc Lelay. A scheduling tool for bridging the gap between aircraft design and aircraft manufacturing. *Proceedings of the International Conference on Automated Planning and Scheduling*, 28(1):347–355, June 2018. doi:10.1609/icaps.v28i1.13910.

24 Brahim Rekiek, Alain Delchambre, Alexandre Dolgui, and Antoneta Bratcu. Assembly line design: A survey. *IFAC Proceedings Volumes*, 35(1):155–166, 2002. 15th IFAC World Congress. doi:10.3182/20020721-6-ES-1901.01647.

25 Emma Rollon and Javier Larrosa. Multi-objective propagation in constraint programming. *Frontiers in Artificial Intelligence and Applications*, 141:128, 2006.

26 Tamara Borreguero Sanchidrián, Tom Portoleau, Christian Artigues, Alvaro García Sánchez, Miguel Ortega Mier, and Pierre Lopez. Exact and heuristic methods for an aeronautical assembly line time-constrained scheduling problem with multiple modes and a resource leveling objective. *hal-03344445*, 2021.

27 Nicolas Schwind and Demirović Emir. Representative solutions for bi-objective optimisation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1436–1443, 2020.

28 Pascal Van Hentenryck, Laurent Michel, Laurent Perron, and Jean-Charles Régin. Constraint programming in opl. In *PPDP*, volume 99, pages 98–116. Springer, 1999.

29 Yeo Keun Kim, Yong Ju Kim, and Yeongho Kim. Genetic algorithms for assembly line balancing with various objectives. *Computers & Industrial Engineering*, 30(3):397–409, 1996. IE in Korea. doi:10.1016/0360-8352(96)00009-5.