# Constraint Programming Model for Assembly Line Balancing and Scheduling with Walking Workers and Parallel Stations

**Xavier Pucel**[1]
ONERA, ONERA DTIS, Toulouse, Université de Toulouse, France

**Stéphanie Roussel**
ONERA, ONERA DTIS, Toulouse, Université de Toulouse, France

## Abstract

In the context of aircraft assembly lines, increasing the production rate and decreasing the operating costs are two important, and sometimes contradictory, objectives. In small assembly lines, sharing production resources across workstations is a simple and efficient way to reduce operating costs. Therefore, workers are not assigned to a unique workstation but can walk between them. On the other side, paralleling workstations is an efficient way to increase the production rate. However, the combination of both strategies create complex conditions for tasks to access the production resources. This paper addresses the problem of allocating tasks to workstations and scheduling them in an assembly line where workers can freely walk across workstations, and where workstations can be organized in parallel. We model this novel problem with Constraint Programming. We evaluate it on real world industrial use cases coming from aircraft manufacturers, as well as synthetic use cases adapted from the literature.

**2012 ACM Subject Classification** Applied computing → Computer-aided manufacturing

**Keywords and phrases** Constraint Programming, Assembly Line, Balancing and Scheduling, Parallel Workstations, Walking Workers

**Digital Object Identifier** 10.4230/LIPIcs.CP.2024.23

**Supplementary Material** *Dataset (Assembly Line Design)*: `https://doi.org/10.57745/EWXS9O` [29]

**Funding** This work was funded by the European Union's NextGenerationEU program, and the national France Relance program. It was coordinated by the Directorate General of Civil Aviation (DGAC) in France.

## 1 Introduction

The production of complex artefacts such as aircraft is commonly performed by the use of assembly lines. In an assembly line, several workstations are dedicated to a subset of assembly tasks, and the artefacts being built travel from one workstation to another until they are complete. As each workstation is always occupied by one instance of the artefact, all instances visit the workstations in the same order, and travel at the same time, after a fixed time interval called the *cycle time*. Since the cycle time is the duration between the completion of two instances of the artefact, its value is of critical importance for business purposes, and a lot of research aims at improving (i.e. minimizing) its value for all sorts of production systems.

A natural way to improve the production rate, without modifying the artifact design, is simply to add more workstations. Since the same amount of work is divided in a larger number of workstations, each workstation has less tasks and complete them faster, thus

[1] Corresponding author

decreasing the cycle time. The main drawback of adding workstations is that it occupies more space in the factory and may not always be possible. Furthermore, even when it is possible, the cost of creating a new workstation can be another obstacle. While in large assembly lines each workstation has its dedicated equipment and operators, this is not necessarily the case in smaller assembly lines, where it is possible for operators to travel, and equipment to be transferred, across workstations. In this context, adding a new workstation can be much cheaper if expensive equipment does not need to be duplicated.

Another way to increase the production rate is to act on the artefact flow, and in particular to extend a production stage with parallel workstations. The requirement of having each task confined to a single workstation then translates to weaker temporal constraints, and it even allows the cycle time to be smaller than the duration of some tasks. It also has the drawback of occupying more space in the factory since more routes must be cleared for the artifacts to travel. Moreover, parallel workstations with shared resources can create complex situations in terms of conflicting access by tasks to resources. These conditions are precisely the highlight of this paper.

In this paper, inspired by an industrial assembly line for small aircraft, we address the problem of increasing the production rate, and use it to evaluate different assembly line structures (number of stages, number of parallel workstations), with a predefined amount of resources. More precisely, our assembly line model has the following features:

- Minimizing the assembly duration.
- The number and layout of workstations is an input of the problem. The layout specifies the number of serial stages, and the number of parallel duplicate workstations in each stage.
- Each workstation has different zones (e.g. inside or outside of the aircraft, front or rear, etc.), and each task occupies one or more zones. Two tasks that occupy the same zone cannot be performed simultaneously.
- Workers can walk freely across workstations. The travel time between workstations is not accounted for, as it is considered negligible with respect to the work time.
- Each worker has one specific skill, and each task requires a given number of workers with each skill (including zero). At each instant a worker can only perform one task, which means tasks compete with one another for accessing workers. As we have walking workers, this competition occurs not only inside workstations, but also across workstations.
- As parallel workstations allow for tasks longer than the cycle time, such a long task can compete with itself for accessing workers. More precisely, two instances of the task performed on two parallel workstations, but shifted by only one cycle, overlap, and thus require twice the amount of workers. This aspect is original, and is explained and illustrated in detail.
- In the problem output, in addition to the smallest possible cycle time, each assembly task is assigned to a unique workstation (or rather a stage in the case of parallel workstations), and is given start and end dates. Moreover, tasks are not preemptive.

We can summarize the contribution of this work as follows:

- we address a novel extension of the Assembly Line Balancing (ALB) problem in which we schedule all tasks, some resources can walk between stations and stations can be parallel;
- we propose a Constraint Programming (CP) model for this problem ;
- we present results associated with data coming from an aircraft manufacturer, which show the applicability of the approach;

- we experiment on public benchmarks that we have adapted to our hypotheses: some corresponding to aircraft assembly lines and the others based on public Resource-Constrained Project Scheduling Problem (RCPSP) instances.

This paper is organized as follows. First the state of the art is reviewed in Section 2, around the ALB and RCPSP variants, and constraint programming in the aeronautics industry. Second, a more detailed account of the assembly line is provided in Section 3, and in particular how parallel workstations behave, and how it impacts the competition for resources between tasks. The CP model is then described in Section 4 along with an illustrative example. Finally, Section 5 relates our experiments on both academic benchmarks and industrial examples. We describe perspectives of this work in Section 6.

## 2 Related Work

The work presented in this paper can be compared with two well known optimisation problems: the Assembly Line Balancing (ALB) problem and the Resource-Constrained Project Scheduling Problem (RCPSP). ALB essentially deals with assigning tasks to workstations [11]. In particular, the Simple Assembly Line Balancing Problem version 2 (SALB2) aims at minimizing the cycle time given a list of tasks, a list of workstations and precedence constraints between tasks. However, in a classical ALB problem, tasks are only assigned to workstations, but are not scheduled, i.e. they are not assigned a start and end date.

A family of variants of ALB addresses assembly lines where some workstations perform the same tasks in parallel. More precisely, the assembly line is composed of a series of *stages*, each stage contains one or several parallel workstations. Each product goes through each stage in order, but only on one workstations at each stage. Using parallel workstations helps decreasing the cycle time, but requires more resources. Parallel ALB approaches minimize either the cycle time or total cost associated to workstations, and assign tasks to workstations, but do not schedule them [12, 3, 4, 16].

In a classical RCPSP, given a list of tasks, precedence constraints between tasks, a list of resources and a consumption relation between tasks and resources, the problem is to assign a start date and an end date to each task so as to minimize the total work time [18]. The main difference with ALB is that the work is not divided among synchronized workstations. However, by reasoning on start and end dates, it supports a precise management of resources. Our approach can be seen as an extension of RCPSP with workstations, work zones in each workstation, and cumulative resources shared across workstations. This is how we could evaluate our approach on the PSPlib benchmark [19].

The RCPSP enjoys a vast amount of variants [15]. Among them is the Multi-Skill RCPSP, in which workers have one or several skills that are required for executing tasks. Our approach does not feature this aspect. Each worker has a unique skill, and a set of $n$ workers with a particular skill can be directly modeled as a renewable resource with a capacity equal to $n$.

There are many extensions of ALB that incorporate RCPSP aspects. Resource Constrained ALB (RCALB) accounts for the resources deployed on each workstation and for how tasks use resources [2, 13, 7], much like RCPSP cumulative resources, or [1] that supports multi-mode RCPSP constraints. In RCALB, resources are assigned to a single workstation, which differs from our approach where resources are shared across workstations.

The variant of Multi-Manned ALB resembles RCALB, except in the case of Walking Workers. These approaches have many common points with ours and are detailed in a dedicated section 2.1. ALB and RCPSP have been combined in other ways described in [8].

Multi-Model ALB models assembly lines where several types of product are manufactured. In this variant, many approaches not only assign tasks to workstations but also schedule them. [24] has a list of tasks with precedence constraints in order to assemble a set of products. Tasks are scheduled on workstations accounting for precedence, equipment, surface and buffers, but completely ignores possible future tasks. In [25] the problem is modified to consider cyclic schedules, in which the list of tasks is expected to be repeated, and the concept of cycle time appears again, event though the assembly line is not pulsated. [21] also computes cyclic schedules for multi-model assembly lines, however the products advance on a treadmill and workstations are not separated and organized into stages.

Some variants of the ALB problem support walking workers. In [32] workers can move from one station to another, and the time to travel between stations is accounted for under a form similar to the Traveling Salesman Problem. However, tasks are not scheduled inside each station. [10] addresses mixed model assembly lines and assigns workers to workstations. However, the assignment of tasks to workstations is predefined as input.

## 2.1 Multi-manned ALB with Walking Workers

An important family of variants of ALBP concerns Multi-manned ALB (MMALBP). Some of these variants model the workers needed to perform tasks in a way similar to how tasks use resources in a RCPSP, much like this paper does. As a consequence, many approaches in this family not only assign tasks to workstations, but also schedule their start and end dates in order to ensure that the tasks assigned to each worker are feasible, both in volume and precedence. In addition, some of these variants have walking workers that travel across workstations. Many of these variants bear significant similarities with our work.

[14] proposes Mixed Integer Linear Programming (MILP) models for MMALB. Each worker is assigned to a station, and each task is assigned to both a workstation and a worker, and is scheduled inside its station. Workers cannot travel to another workstation, and working zones and parallel workstations are not modelled.

[30] addresses MMALBP with walking workers. Each worker can travel to a predefined set of stations (possibly all), and each station can host several workers at once. Each task is assigned to a station and a worker, and is scheduled inside its station. It only differs with this work in that working zones, and more importantly parallel workstations, are not accounted for.

[23] describes MMALB with moving workers. Each task has a set of compatible workers, which models worker skills, and a set of mutually exclusive tasks, which could be used to model working zones. Each workstation is divided into sides, and each worker is assigned to at most two sides of a single workstation. Each task requires one or several sides, and is assigned to a single station, as well as one worker per required side. Tasks are not scheduled by date, but a partial ordering is computed during optimization. Workers only travel across sides of a single workstation; furthermore parallel workstations are not modelled in this paper.

[22] addresses MMALB in production lines where the product does not advance in pulses from one station to another, but rather advances continuously as on a treadmill. Tasks are assigned to workers and scheduled inside an envelope called “cycle time” so as to occupy a bounded region of the treadmill. However, there are no global date boundaries like in pulsated assembly lines.

[9] defines variable workplaces ALB, in which each workstation has a list of so-called “mounting positions”, that are gathered into workplaces in a possibly different way at each workstation. Tasks are assigned to workplaces that must contain the required mounting

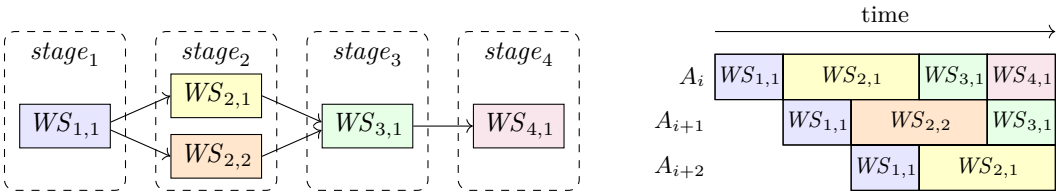



**Figure 1** An example of an assembly line with 4 stages and two parallel workstations in the second stage. The left side depicts the physical layout of the assembly line. The right side depicts the temporal schedule of several consecutive aircraft. Note that aircraft $A_i$ and $A_{i+2}$ go through workstation $WS_{2,1}$ while $A_{i+1}$ goes through workstation $WS_{2,2}$. This allows each aircraft to spend 2 cycles in stage 2 without conflict.

positions. Two workplaces cannot overlap on the same workstation. In that work, tasks are scheduled inside workstations, and it accounts for extra-long tasks (longer than one cycle time) by splitting them into smaller tasks.

## 2.2 ALB and RCPSP Applied to Aircraft Manufacturing

The application of operation research techniques to aircraft manufacturing is not novel. In [31] the problem of scheduling assembly tasks is formulated as a RCPSP problem. Similarly, [5] addresses the problem of assigning operators and models the assembly line as a RCPSP. [17] plans the resources for aircraft turnaround operations as an RCPSP problem. However, the concept of workstation is absent from those studies.

In a more closely related approach, [26] both allocates tasks to workstations and schedules their executions dates. The model accounts for renewable resources and work zones and scales up to a large number of tasks. [27] addresses multi-criteria optimization for the preliminary design of assembly lines. Tasks are both assigned to workstations and scheduled inside them, and work zones and resources such as machines are accounted for. It introduces specific types of constraints such as neutralized zones, and skill exclusion. A simplified version of their use-case was used in the 2023 XCSP3 competition [6]. While they do not account for parallel workstations, in this paper we modify their use-case in the evaluation section.

# 3 Aircraft Assembly Line Description

The production systems involved in aircraft construction are complex as they are usually composed of several factories that can produce one part of the aircraft or assemble several parts together. Each factory has several key areas: storage, offices, preparation areas, assembly lines, locker rooms, etc. In this work, we focus on the final assembly line of a manufacturer of small aircraft. This final assembly line is in charge of assembling the fuselage, the wings, the propeller and the engine, and of installing electrical and hydraulic systems in the aircraft.

As done classically in the aeronautics industry, the assembly line we consider here is composed of several workstations, and is pulsated. In a sequential assembly line, each aircraft stays a given duration in each workstation before going to the next one or going out of the line after the final workstation. This duration, called *cycle time*, and noted $Ct$, is imposed on the manufacturer: it is calculated so as to guarantee a certain number of aircraft delivered per

month or per year[2]. A set of assembly tasks on the aircraft is performed on each workstation. Tasks are not pre-emptive, i.e. no task can be interrupted, and they cannot overlap several workstations. In fact, no assembly task can be performed while the aircraft moves from one workstation to the next.

In order to be executed, tasks can require equipment and operators with particular skills. Each of them is only available in limited quantity, and we represent it through an abstract cumulative renewable resource. All tools and operators are shared by all the workstations of the assembly line. If a task requires a shared resource in a given workstation, it impacts the resource's available capacity on all other workstations during the task execution. For instance, if a resource with capacity 1 is required for the first task of the first workstation, and if this task has a duration equal to 2, then the resource is not available during the two first time units of each cycle in the entire assembly line.

The aircraft is divided into several zones in which tasks are performed. Two tasks that occupy the same zone cannot be performed simultaneously on the same workstation. Zones are represented as renewable resources of capacity 1; however, unlike equipment and operators, each workstation has its own work zones. This means that two tasks that occupy the same zone, need to be scheduled either one after another on a single workstation, or on different workstations. Note that it would be possible to consider zones with capacity greater than 1.

The assembly line considered in this paper features duplicated workstations that operate in parallel. It is organised as a series of *stages*, each stage being composed of one or several identical workstations that operate in parallel. This allows the aircraft to stay longer in these duplicated workstations. Stages with one workstation behave much like in sequential assembly lines. However, in a stage with $n$ parallel workstations, each aircraft stays $n$ times the cycle time on its workstation before heading to the next stage. At each cycle, one aircraft goes out of each stage and into the next one. For instance, in the assembly line illustrated in Figure 1, the second stage contains two workstations. When entering the second stage, each aircraft $A_i$ goes either on workstation $WS_{2,1}$ or workstation $WS_{2,2}$ and stays there for a duration equal to $2 \cdot Ct$ in the example. As $WS_{2,1}$ and $WS_{2,2}$ are shifted of one cycle time, there is still an aircraft entering $WS_{3,1}$ at each cycle.

Parallel duplicate workstations allow the aircraft to stay in a stage for a duration greater than the cycle time. This allows for tasks longer than the cycle time to be realized. In the example of Figure 1, stages 1, 3 and 4 last one cycle time, but stage 2 lasts two cycles, so it can be assigned a task of a duration up to $2 \cdot Ct$.

Note that while a stage is physically, or spatially, decomposed into workstations, it would be misleading to divide the time period an aircraft spends on a stage into time periods called "workstations". For example in Figure 1, while stage 2 physically has 2 workstations, and while aircraft $A_i$ stays two cycles in workstation $WS_{2,1}$, it would be incorrect to refer to the first half of this time period stay as "the first workstation of stage 2", since aircraft $A_i$ will never travel to workstation $WS_{2,2}$. Instead, we say that stage 2 is temporally decomposed into two *substages*, and stages 1, 3 and 4 are composed of one substage.

To summarize, all substages have the same duration of 1 cycle. The assembly line has as many substages as it has workstations. At each instant, each workstation is occupied by exactly one aircraft, and *each aircraft is in a different substage of its construction.* Each aircraft goes through all construction substages in the same order, however it does not necessarily travel through all workstations.

[2] More precisely, the long term production rate, or *takt*, is an objective imposed on the manufacturer. Our work evaluates whether a factory design can reach an operational *cycle time* consistent with the takt under realistic conditions.

| Task | Duration | Zone 1 | Zone 2 | Resource 1 | Resource 2 |
|---|---|---|---|---|---|
| $t_1$ | 2 | × | | 2 | 1 |
| $t_2$ | 8 | × | | 1 | |
| $t_3$ | 6 | | × | | |
| $t_4$ | 2 | | × | | 1 |
| $t_5$ | 2 | × | × | | |
| $t_6$ | 4 | × | | 1 | 1 |
| $t_7$ | 4 | | × | | 1 |
| | | | | $capa_1 = 3$ | $capa_2 = 2$ |

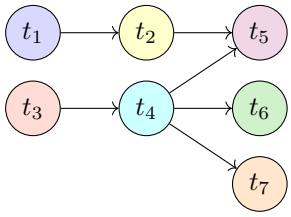
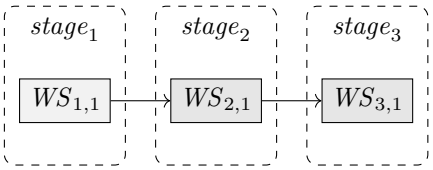
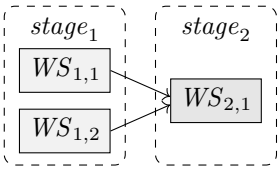



**Figure 2** Illustration of the simple assembly line problem detailed in Example 2.




**(a)** First layout, denoted $\mathsf{L}_1$.




**(b)** Second layout, denoted $\mathsf{L}_2$.

**Figure 3** Illustration of two possible layouts for the problem described in Example 2.

Parallel workstations are a useful tool to increase the production rate. They make it possible to schedule tasks that last more than the cycle time, or conversely to increase the production rate so that the cycle time is smaller than the longest assembly task, which is impossible on sequential assembly lines. The drawbacks of parallel workstations are an increased surface occupation, increased tools and operators requirements, and an increased number of aircraft under construction, which can be a significant financial constraint.

When combined with walking workers, parallel workstations can create counter-intuitive situations, where a task seems to use several times its amount of workers. This happens on stages with duplicated workstations, that are assigned a task longer than the cycle time. Assume in Figure 1 that a task of length $1.5 \cdot Ct$ is executed at the start of stage 2, and in theory only requires 1 worker. When aircraft $A_i$ arrives on workstation $WS_{2,1}$ the task starts and one worker starts working on it, for the next $1.5 \cdot Ct$. One $Ct$ later, aircraft $A_{i+1}$ arrives on workstation $WS_{2,2}$, but the worker is still busy on the aircraft $A_i$. As a consequence, a task that would only require one worker in a sequential layout, will in fact need two workers in the layout of Figure 1 and with a cycle time shorter than its duration. This aspect is illustrated later in Example 2 and Figure 5.

Our goal is to study the impact of paralleling workstations on the production rate. To that end, we model the assembly line under several layout assumptions and minimize the cycle time under constant resource capacity assumptions.

## 4 Assembly Line Model

This section introduces definitions for our multi-manned assembly line balancing problem with walking workers and parallel stations. The elements constituting such a problem and an associated solution are described first. Then, a constraint programming model in the OPL ([20]) language is detailed, explained and illustrated.

## 4.1 Formal Definitions

▶ **Definition 1** (MALBWP). *A Multi-manned Assembly Line Balancing problem with Walking workers and Parallel stations is a tuple* $(\mathcal{T}, nS, \mathcal{Z}, \mathcal{R}, Prec, Occ, Cons)$ *where:*

- $\mathcal{T}$ *is a set of assembly tasks. Each task* $t \in \mathcal{T}$ *is associated with its duration* $dur_t$.
- $nS$ *is the number of stages.* $\mathcal{S}$ *denotes the set* $[1..nS]$ *of assembly stages, where each stage can have one or more parallel workstations. For each stage* $s \in \mathcal{S}$*, its number of parallel workstations (and thus substages) is noted* $w_s$*. The total number of substages is noted* $W = \sum_{s \in \mathcal{S}} w_s$.
- $\mathcal{Z}$ *is a set of work zones representing the different areas of the aircraft, in which workers can perform different tasks.*
- $\mathcal{R}$ *is a set of resources or tools that are needed to perform certain tasks. Each resource* $r \in \mathcal{R}$ *is associated with its capacity* $capa_r$.
- $Prec \subseteq \mathcal{T} \times \mathcal{T}$ *is the precedence relation between tasks. For each pair of tasks* $(t_1, t_2) \in Prec$*,* $t_2$ *cannot start until* $t_1$ *is finished.*
- $Occ \subseteq \mathcal{T} \times \mathcal{Z}$ *is the occupation relation between tasks and zones.* $(t, z) \in Occ$ *means that task* $t$ *occupies zone* $z$*. Each task can use any number of zones.*
- $Cons \subseteq \mathcal{T} \times \mathbb{N} \times \mathcal{R}$ *is the resource usage relation between tasks and resources.* $(t, n, r) \in Cons$ *means that task* $t$ *uses* $n$ *units of resource* $r$*. Each task can use any number of resources. For each resource* $r$*, we denote* $Cons_r$ *the set* $\{(t, n) | (t, n, r) \in Cons\}$*, i.e. the weighted set of tasks that consumes* $r$.

This model features two different types of resources. Zones have capacity 1, are specific to each workstation, and are present in all workstations. Resources have a finite capacity (possibly 1), however they are shared across all workstations. Note that this type of problem could easily be extended to workstation-specific resources with finite capacity as in [26].

▶ **Example 2.** Figure 2 describes a small multi-manned assembly line balancing problem with walking workers and parallel stations. The problem contains 7 tasks, 2 zones and 2 resources. The zone occupation relation, the resource consumption relation and the resource capacities are detailed in the table on the left, and the graph induced by the precedence relation is depicted on the right. Resource 1 has capacity 3, and resource 2 has capacity 2.

Figure 3 illustrates two possible layouts for the assembly line with 3 workstations, i.e. $W = 3$. In layout $L_1$ (Figure 3a) the stage set $S = \{1, 2, 3\}$ contains 3 stages of 1 workstation each, i.e. $w_1 = w_2 = w_3 = 1$. In layout $L_2$ (Figure 3b), there are two stages $S = \{1, 2\}$, and the first one is composed of two workstations, i.e. $w_1 = 2$ and $w_2 = 1$.

▶ **Definition 3** (Solution). *A solution to a given MALBWP is a tuple* $(Ct, start)$ *defined as follows.*

- $Ct$ *is the pulse rate of the assembly line.*
- *The duration of stage* $s \in \mathcal{S}$ *equals* $w_s.Ct$*, the first stage starts at time* 0 *and each other stage starts when its predecessor stage ends. The start date of stage* $s$ *is denoted* $\text{stageStart}_s$ *and its end date is denoted* $\text{stageEnd}_s$.
- $start$ *associates a start date to each task* $t \in \mathcal{T}$*, noted* $start_t$*. The end date of each task, noted* $end_t$ *is the sum of its start date and its duration.*
- *Each task is performed inside a unique stage, i.e. for each task* $t \in \mathcal{T}$*, there exists a unique stage* $s \in \mathcal{S}$ *such that* $\text{stageStart}_s \leq start_t \leq end_t \leq \text{stageEnd}_s$.
- *Two tasks that occupy the same zone do not overlap temporally. Formally, if* $(t_1, z_1) \in Occ$ *and* $(t_2, z_2) \in Occ$*, we have either* $t_1 = t_2$*,* $z_1 \neq z_2$*,* $end_{t_1} \leq start_{t_2}$*, or* $end_{t_2} \leq start_{t_1}$.

- *At each instant, resources are not used beyond their capacity* across all workstations. *Formally, we denote* $consumption(r, \tau)$ *the amount of units of resource* $r$ *used at instant* $\tau$*, and require that at each instant* $\tau \in [0, Ct]$ *and for each resource* $r \in \mathcal{R}$*, we have* $consumption(r, \tau) \leq capa_r$ *where:*

$$consumption(r, \tau) = \sum_{(t,n) \in Cons_r} \sum_{k=1}^{W} \begin{cases} n & \text{if } start_t \leq \tau + k.Ct \leq end_t \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The last constraint on the consumption of resources across workstations is, to our knowledge, original. At each instant, there is one aircraft in each assembly substage, thus each resource is used simultaneously by each aircraft. Hence, in Equation (1), in the second summation, $k$ ranges across all substages ($W$ is the total number of substages). Moreover, in a stage with multiple workstations, a task may last longer than one $Ct$. In this case, this task is performed simultaneously on multiple workstations, and contributes to the resource usage multiple times. This is captured by the $\tau + k.Ct$ term of Equation (1), and illustrated in the example that follows.

▶ **Example 4.** Figures 4 and 5 depict two solutions for the problem of Example 2 respectively associated with layout $L_1$ and layout $L_2$.

With layout $L_1$ (Figure 4), the best possible cycle time is 8. Zone occupation and resource usage can be visualized by the intervals depicted in the figure. Since each aircraft instance has its own work zones, each zone is duplicated in each stage and substage. On the opposite, resources are shared between all assembly stages, and are used simultaneously. Thus, the aircraft in stage 1 consumes 2 units of resource 1, at the same time as the aircraft in stage 2 consumes 1 unit of it. The same holds for all stages, resources and tasks. This is illustrated on the right hand side of Figure 4.

In Figure 5, the stage set $S = \{1, 2\}$ contains two stages, however the first stage contains two parallel workstations and therefore two substages. This means we have $w_1 = 2$, and $w_2 = 1$, which allows task $t_2$ to start at date 2 in substage 1.1 and end at date 8 in substage 1.2 (as tasks cannot span across stages, but can span across substages). In this setting, the smallest cycle time is 6.

A feature specific to our approach is how task $t_2$ consumes resource 1 "several times" in Figure 5. At each instant, 3 aircraft are being assembled, one in each assembly stage. Let us consider the moment when 3 time units have elapsed in the current cycle and that aircraft $A_i$, $A_{i-1}$ are in stage 1, and aircraft $A_{i-2}$ in stage 2. $A_i$ has already spent 3 time units in stage 1, meaning that 1 time unit of task $t_2$ has already been executed for it. Aircraft $A_{i-1}$ is also in stage 1 but has entered it $Ct + 3 = 9$ time units ago and is therefore in substage 1.2. Task $t_2$ is not yet finished for this aircraft, as 7 time units have already been executed. Consequently, two instances of the task $t_2$ are simultaneously executed on two different aircraft instances, thus using twice the amount of resource 1, as illustrated on the right. Note that at the same time, task $t_6$ is executed on aircraft $A_{i-1}$ in stage 2, consuming an additional unit of resource 1.

## 4.2 Constraint Programming Representation

The MALBWP problem can be represented in CP provided that an upper bound on the pulse time $maxCt$ is added to the problem inputs. The resulting temporal horizon considered, denoted $H$, is equal to $\sum_{s \in \mathcal{S}} w_s \cdot maxCt$.

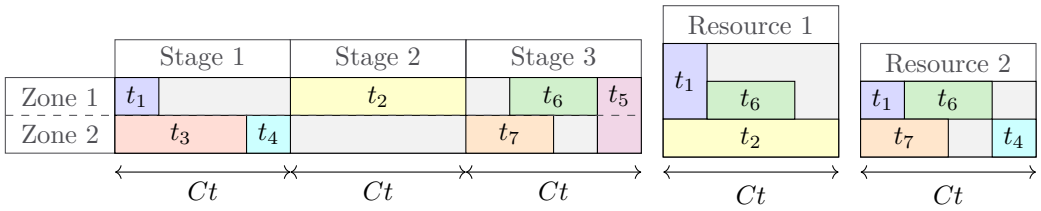



**Figure 4** Solution with $Ct = 8$ for the toy problem with layout $L_1$.

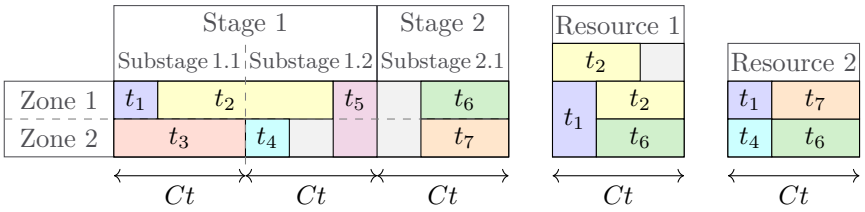



**Figure 5** Solution with $Ct = 6$ for the toy problem with layout $L_2$.

Our representation makes use of the OPL constraint programming language [33], as it makes heavy use of its concept of interval variables and associated constraints, in particular for optional intervals. An interval variable has a range $[min, max]$ that specifies its earliest start date and latest end date, and it can be optional, meaning in that case that it is not necessarily present in the produced schedule. The OPL functions startOf, endOf, and lengthOf return respectively the start date, end date and length of a given interval.

Our constraint model uses the following decision variables. All interval variables have the range $[0, H]$ unless explicitly mentioned.

- The rate is modeled by an integer variable **Ct** in the range $[0, maxCt]$.
- For each task $t \in \mathcal{T}$, the mandatory interval variable $\mathbf{itv}_t$ represents the execution of task $t$. Its duration is fixed to $dur_t$.
- For each task $t \in \mathcal{T}$ and each stage $s \in \mathcal{S}$, the optional interval variable $\mathbf{itv}_{t,s}$. If present, this interval has the fixed duration $dur_t$, and represents the execution of task $t$ in stage $s$. If absent, it means task $t$ is executed in another stage.
- For each task $t \in \mathcal{T}$, each stage $s \in \mathcal{S}$ and each substage $w \in [1..w_s]$, the optional interval variable $\mathbf{itv}_{t,s,w}$ represents the execution of task $t$ during the $w$-th substage of stage $s$. If absent, it means that task $t$ is executed on another stage or substage. For example in Figure 5, task $t_1$ is executed during the first substage of the first stage. Hence $\mathbf{itv}_{t_1,1,1}$ is present, and both $\mathbf{itv}_{t_1,1,2}$ and $\mathbf{itv}_{t_1,2,1}$ are absent. On the other hand, task $t_2$ is executed in both substages of stage 1, so $\mathbf{itv}_{t_2,1,1}$ and $\mathbf{itv}_{t_2,1,2}$ are both present and $\mathbf{itv}_{t_2,2,1}$ is absent.
- For each task $t \in \mathcal{T}$, each stage $s \in \mathcal{S}$ and each substage $w \in [1..w_s]$, the optional interval variable $\mathbf{itvRsc}_{t,s,w}$ has a range equal to $[0, maxCt]$. It represents the time during which the execution of time $t$ on the $w$-th substage of station $s$ uses its resources. If task $t$ is executed on another stage, or not during the $w$-th substage, this interval is absent. As for the previous set of interval variables, $\mathbf{itvRsc}_{t_1,1,1}$, $\mathbf{itvRsc}_{t_2,1,1}$ and $\mathbf{itvRsc}_{t_2,1,2}$ are present, whereas $\mathbf{itvRsc}_{t_1,2,1}$ and $\mathbf{itvRsc}_{t_2,2,1}$ are absent.

The constraints of our encoding are as follows.

## Assembly Line Expressions

We start by defining some expressions for the start and end dates for each stage, and for each pulse cycle inside each stage. Expressions are neither constraints nor decision variables, but are useful for efficiently expressing recurrent patterns in constraints.

$$\forall s \in \mathcal{S}, \quad \text{stageStart}_s = \sum_{i=1}^{s-1} w_i \cdot \mathbf{Ct} \tag{2}$$

$$\forall s \in \mathcal{S}, \quad \text{stageEnd}_s = \sum_{i=1}^{s} w_i \cdot \mathbf{Ct} \tag{3}$$

$$\forall s \in \mathcal{S}, \forall w \in [1..w_s], \quad \text{subStageStart}_{s,w} = \text{stageStart}_s + (w-1) \cdot \mathbf{Ct} \tag{4}$$

$$\forall s \in \mathcal{S}, \forall w \in [1..w_s], \quad \text{subStageEnd}_{s,w} = \text{stageStart}_s + w \cdot \mathbf{Ct} \tag{5}$$

Equations (2) and (3) define expressions for the start and end dates of each stage. These dates directly depend on the value of **Ct**. Similarly, Equations (4) and (5) define expressions for the start and end date of every substage.

Note that $\text{stageStart}_1 = 0$, which means the first stage starts at time 0. Moreover, $\text{stageStart}_{s+1} = \text{stageEnd}_s$ for all stages except for the last one, which means each stage starts right when its predecessor ends. Finally, a stage with $w_s$ workstations has a duration of $w_s \cdot Ct$, and each substage has a duration of exactly one $Ct$.

## Stage and Zone Constraints

The stage constraints express that each task can only take place inside one stage. Furthermore, two tasks that occupy the same zone cannot be executed at the same time in the same stage. This set of constraints features the alternative OPL keyword, that accepts one interval $a$ and one set of optional intervals $B$ as arguments, and has the following semantics. If $a$ is absent, then all intervals in $B$ are absent. If $a$ is present, then exactly one interval is $B$ is present, and it has the same start and end dates as $a$.

$$\forall (t,t') \in Prec, \quad \mathsf{endBeforeStart}(\mathbf{itv}_t, \mathbf{itv}_{t'}) \tag{6}$$

$$\forall z \in \mathcal{Z} \quad \mathsf{noOverlap}\big(\{\mathbf{itv}_t | (z,t) \in Occ\}\big) \tag{7}$$

$$\forall t \in \mathcal{T}, \quad \mathsf{alternative}\Big(\mathbf{itv}_t, \big\{\mathbf{itv}_{t,s} | s \in [1..nS]\big\}\Big) \tag{8}$$

$$\forall t \in \mathcal{T}, \forall s \in \mathcal{S}, \quad \text{stageStart}_s \leq \mathsf{startOf}(\mathbf{itv}_{t,s}, H) \tag{9}$$

$$\forall t \in \mathcal{T}, \forall s \in \mathcal{S}, \quad \mathsf{endOf}(\mathbf{itv}_{t,s}, 0) \leq \text{stageEnd}_s \tag{10}$$

Constraints (6) expresses that two tasks with a precedence constraint must be scheduled in the proper order. Constraints (7) forbid two tasks that occupy the same zone to be executed simultaneously in the same stage. Constraints (8) force interval $\mathbf{itv}_t$ to equal exactly one of the $\mathbf{itv}_{t,s}$. Constraints (9) and (10) ensure that each interval $\mathbf{itv}_{t,s}$, if present, takes place during stage $s$. The second arguments to the startOf and endOf functions (respectively $H$ and 0) are default values in case the interval is absent. In conjunction to Constraints (8), this enforces every task to be executed inside one stage.

## Substage Constraints

We consider here constraints associated with the execution of tasks in substages.

$$\forall t \in \mathcal{T}, \forall s \in [1..nS], \qquad \mathsf{span}\Big(\mathbf{itv}_{t,s}, \big\{\mathbf{itv}_{t,s,w} | w \in [1..w_s]\big\}\Big) \tag{11}$$

$$\forall t \in \mathcal{T}, \forall s \in \mathcal{S}, \forall w \in w_s, \qquad \text{subStageStart}_{s,w} \leq \mathsf{startOf}(\mathbf{itv}_{t,s,w}, H) \tag{12}$$

$$\forall t \in \mathcal{T}, \forall s \in \mathcal{S}, \forall w \in w_s, \qquad \mathsf{endOf}(\mathbf{itv}_{t,s,w}, 0) \leq \text{subStageEnd}_{s,w} \tag{13}$$

$$\forall t \in \mathcal{T}, \forall s \in \mathcal{S}, \qquad \mathsf{lenghtOf}(\mathbf{itv}_{t,s}) = \sum_{w \in [1..w_s]} \mathsf{lengthOf}(\mathbf{itv}_{t,s,w}) \tag{14}$$

Constraints (11) uses the span OPL constraint, that accepts an interval $a$ and a set of optional intervals $B$ as arguments, and has the following semantics. If $a$ is absent, then all intervals in $B$ are absent. If $a$ is present, then at least one interval in $B$ is present and the start (resp. end) date of $a$ equals the earliest start date (resp. latest end date) of all the present intervals in $B$.

Constraints (12) and (13) ensure that each interval $\mathbf{itv}_{t,s,w}$, if present, lies within the $w$-th substage. As a consequence, for each task $t$ and stage $s$, two intervals $\mathbf{itv}_{t,s,w}$ and $\mathbf{itv}_{t,s,w'}$ cannot overlap, unless $w = w'$.

Constraints (14) ensure that the cumulated length of all present $\mathbf{itv}_{t,s,w}$ intervals equals the length of their spanning $\mathbf{itv}_{t,s}$ interval. Since they cannot overlap because of Constraints (12) and (13), this ensures that the intervals $\mathbf{itv}_{t,s,w}$ entirely cover the interval $\mathbf{itv}_{t,s}$ without any gaps, for each task $t$ and stage $s$.

## Resource Constraints

As illustrated in Example 2, resource usage is a pattern that is repeated every cycle. In order to model resource usage across workstations with interval variables, we rewrite the consumption of resources of Equation (1) as a consumption elements set. Formally, for each resource $r$, we consider the set $\mathcal{C}_r$ of consumption elements $\{(\sigma, \tau, n)\}$ where $\sigma$ is the start date of the consumption, $\tau$ is its end date and $n$ the number of units it consumes. $\mathcal{C}_r$ is built following Algorithm 1, where % is the integer remainder.

**Algorithm 1** Computation of $\mathcal{C}_r$ for each resource $r$.

```
1:  function computeCr(r)
2:      C_r ← ∅
3:      for (t, n) ∈ Cons_r do
4:          if (dur_t ≤ Ct) ∧ (start_t % Ct + dur_t ≤ Ct) then
5:              C_r ← C_r ∪ {(start_t % Ct, end_t % Ct, n)}
6:          else
7:              before ← Ct − start_t % Ct
8:              after ← end_t % Ct
9:              q ← ⌊(dur_t − before − after) / Ct⌋
10:             C_r ← C_r ∪ {(start_t % Ct, Ct, n), (0, end_t % Ct, n), (0, Ct, n · q)}
        return C_r
```

$\mathcal{C}_r$ is initialized to the emptyset. Then, for each tuple $(t, n) \in Cons_r$, we add one or several elements to $\mathcal{C}_r$. If the task is contained in a unique substage (Line 4), then we shift the task so that it is contained in the interval $[0, Ct]$. Otherwise, we consider the duration of the task in its first substage (Line 7) and in its last substage (Line 8). Then, the quotient

$q$ (Line 9) denotes the number of substages over which the task completely spans. We finally add consumptions corresponding to the first substage, to the last and all the included substages, and shift them so that they belong to the interval $[0, Ct]$.

The OPL encoding uses the $\mathbf{itvRsc}_{t,s,w}$ intervals to represent the tuples from the $\mathcal{C}_r$ set for each resource. The integer remainder operator can be eluded thanks to the $\mathbf{itv}_{t,s,w}$ intervals.

$$\forall t \in \mathcal{T}, \forall s \in \mathcal{S}, \forall w \in w_s, \quad \mathsf{endOf}(\mathbf{itvRsc}_{t,s,w}, 0) \leq \mathbf{Ct} \tag{15}$$

$$\forall t \in \mathcal{T}, \forall s \in \mathcal{S}, \forall w \in w_s, \quad \mathsf{presenceOf}(\mathbf{itvRsc}_{t,s,w}) = \mathsf{presenceOf}(\mathbf{itv}_{t,s,w}) \tag{16}$$

$$\forall t \in \mathcal{T}, \forall s \in \mathcal{S}, \forall w \in w_s, \quad \mathsf{startAtStart}(\mathbf{itvRsc}_{t,s,w}, \mathbf{itv}_{t,s,w}, \text{subStageStart}_{s,w}) \tag{17}$$

$$\forall t \in \mathcal{T}, \forall s \in \mathcal{S}, \forall w \in w_s, \quad \mathsf{endAtEnd}(\mathbf{itvRsc}_{t,s,w}, \mathbf{itv}_{t,s,w}, \text{subStageStart}_{s,w}) \tag{18}$$

$$\forall r \in \mathcal{R}, \quad \sum_{(t,n) \in Cons_r} \mathsf{pulse}(\mathbf{itvRsc}_{t,s,w}, n) \leq capa_r \tag{19}$$

Constraints (15) ensure that resource usage intervals fall into the $[0, \mathbf{Ct}]$ time frame, since they represent the use of each resource at each substages. Constraints (16) impose that the only resource usage intervals that are present are those that correspond to a present substage interval for this task. Constraints (17) and (18) use the startAtStart and endAtEnd OPL constraints with delay. When both intervals are present they are respective shorthands for $\mathsf{startOf}(\mathbf{itvRsc}_{t,s,w}) + \text{subStageStart}_{s,w} = \mathsf{startOf}(\mathbf{itv}_{t,s,w})$, and $\mathsf{endOf}(\mathbf{itvRsc}_{t,s,w}) + \text{subStageStart}_{s,w} = \mathsf{endOf}(\mathbf{itv}_{t,s,w})$, and do nothing if at least one interval is absent.

The final constraints (19) use the pulse OPL function, that represents a cumulative function. The pulse primitive accepts an interval $a$ and an integer value $h$, and describes the function with value $h$ in interval $a$ (if present), and 0 elsewhere. Constraints (19) ensure that the resources consumed by all tasks at some time during the range $[0, Ct]$ do not exceed the resource capacity.

Figure 6 illustrates the relationship between all interval variables used in the CP model. Absent intervals are grayed, mandatory intervals have a thick border. A dotted line between two intervals indicates that they are connected by the constraints indicated on the right. This figure represents a $[3, 1]$ layout, i.e. a layout with 3 workstations in the first stage and one in the second stage, and a single task $t$ that uses $n$ units of a resource $r$ with capacity $3n$. Interval $\mathbf{itv}_t$ ranges across the whole planning horizon. Each interval $\mathbf{itv}_{t,s}$ ranges across stage $s$. Each interval $\mathbf{itv}_{t,s,w}$ ranges across substage $s, w$. Intervals $\mathbf{itvRsc}_{t,s,w}$ all range across $[0, \mathbf{Ct}]$.

## 5 Experimentation

This section presents experimental results for the constraint model presented in this paper. Experiments were all run using IBM CP Optimizer 20.1.0 through the Java API on a 20-core Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz with 62GB RAM. We first present results obtained on PSPLib-based benchmarks, then results obtained on two real assembly line datasets.

### 5.1 Results for PSPLib-based Benchmarks

Our model can be seen as an extension of a classical RCPSP problem $(\mathcal{A}, \mathcal{R}es, \mathcal{C}ons, \mathcal{P}rec)$, where $\mathcal{A}$ is the set of activities, $\mathcal{R}es$ the set of resources, $\mathcal{C}ons$ the consumption relationship and $\mathcal{P}rec$ the precedence relationship. Given a set of stages $\{1..nS\}$ and their associated

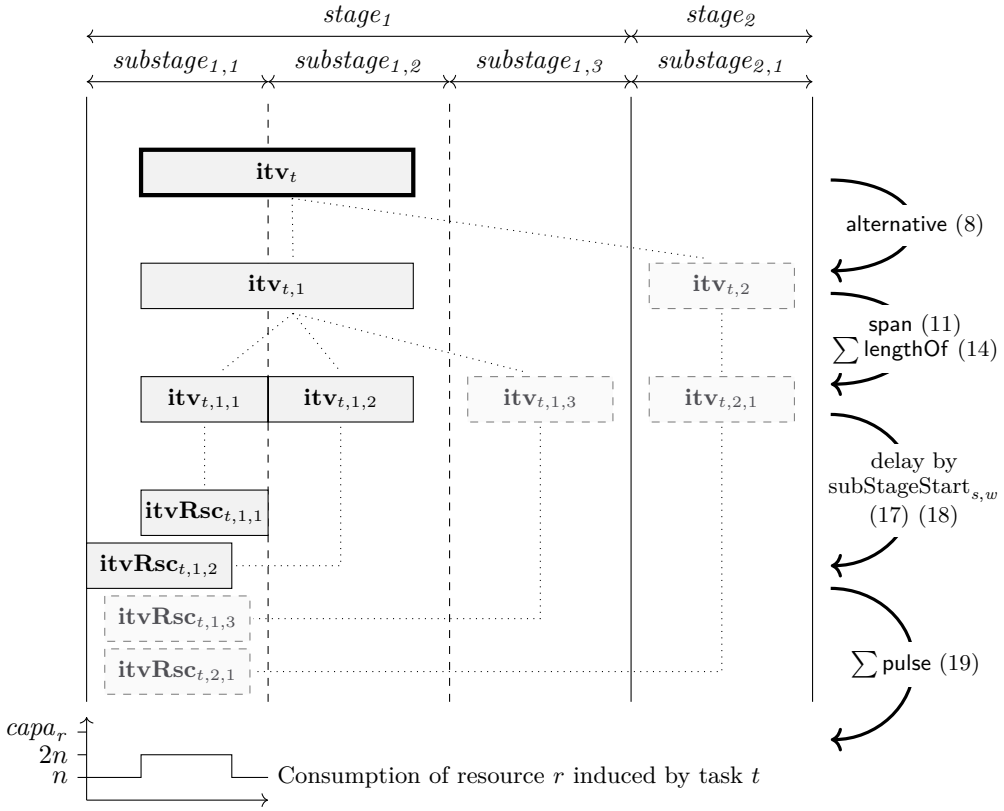



**Figure 6** Illustration of the interval variables hierarchy in the CP model.

number of workstations, and a RCPSP problem, we consider the associated MMALBWP $(\mathcal{A}, nS, \emptyset, \mathcal{R}es, \mathcal{P}rec, \emptyset, \mathcal{C}ons)$ in which the set of zones is empty. The resulting dataset will be made public upon acceptance of the paper.

We adapted problems from the PSPLib [19], converting them to MMALBWP by adding a stage size specification, and used a time limit of 5 minutes. The results on the 150 first problems of the `j30rcp` benchmark, detailed in appendix A, demonstrate several results summarized in Table 1a, and in graph in Table 1b that counts the number of instances where some layouts *strictly* improve other layouts. Below are additional considerations.

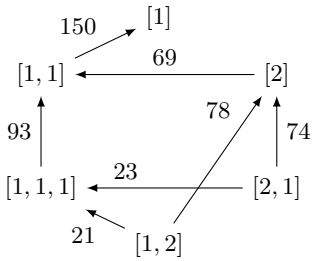
- With one workstation, the cycle time equals the RCPSP makespan.
- More workstations always means a better or equal cycle time (in sequence or in parallel).
- Adding a second workstation, in sequence (layout $[1, 1]$) or in parallel (layout $[2]$), always strictly improves the cycle time of layout $[1]$.
- Adding a third workstation further improves the cycle time in 62% of sequential instances ($93/150$, $[1, 1, 1]$ compared to $[1, 1]$), and in 51% of parallel instances ($(74 + 78)/300$, $[2, 1]$ and $[1, 2]$ compared to $[2]$).
- At constant number of workstations, merging two stages yields a better cycle time in 25% of instances ($(69 + 23 + 21)/450$, $[2]$ compared to $[1, 1]$ plus $[2, 1]$ and $[1, 2]$ compared to $[1, 1, 1]$). This lower number is explained by the fact that resources become a limiting factor.
- Adding stages makes the problem more difficult for the solver, especially with parallel workstations.

**Table 1** Statistics of the solutions of the first 150 instances of the `j30rcp` PSPLIB benchmark.

**(a)** Results summary.

| Layout | #Solved | #Optimal | Avg. Time (s) |
|---|---|---|---|
| [1] | 150 | 150 | 2.5 |
| [1, 1] | 150 | 122 | 72.3 |
| [2] | 134 | 99 | 124.1 |
| [1, 1, 1] | 150 | 84 | 151.1 |
| [2, 1] | 149 | 59 | 193.9 |
| [1, 2] | 150 | 64 | 184.5 |

**(b)** Number of instances in which a layout yields a strictly better cycle time than another.




## 5.2 Industrial Assembly Line

This work was initially motivated by the final assembly line of a manufacturer of small aircraft, who already uses a parallel workstation.

The assembly line model contains 51 tasks, 93 precedence constraints, 12 work zones, and 23 resources. Tasks use between 1 and 4 resource units. We have tested two factory layouts: the first one has 5 sequential stages of one workstation each ($[1, 1, 1, 1, 1]$); the second layout duplicates the first workstation in parallel ($[2, 1, 1, 1]$). We solved the model with a 30 minutes timeout. The solver finds its best solution in a few seconds, but fails to prove that the solution is optimal in the rest of its computation time. The results, and in general the entire modeling work, helped analyse and consolidate the different factory configurations. Note that due to confidentiality issues, we do not provide more information about this dataset nor the obtained results.

## 5.3 Assembly Line Design Use Case

We tested our approach on the assembly line preliminary design problem originally dataset [28] presented in [27] and adapted for the 2023 XCSP3 competition [6]. In the latter, the assembly line is composed of 4 stages with one workstation each. The objective is to minimize the number of operators in the line. We adapt these benchmarks by fixing the number of available operators, ignoring neutralization constraints, and optimizing the cycle time when considering several factory layouts. We have considered 7 layouts for each of the three instances, resulting in a dataset of 21 instances available online [29].

These instances are larger and much more challenging than the ones presented before. We gave a 10mn time limit to the solver, it only managed to find a solution for 15 instances out of 21, and it still failed to prove the optimality of all instances, as reported in Table 2. Furthermore, in instance 3, the cycle time found for stage sizes $[1, 3]$ (i.e. two stages of respectively 3 and 1 workstations) is higher than the one found for stage sizes $[1, 2, 1]$, which indicates that the solver failed to find the same solution within its time limit.

# 6 Conclusion

This paper presents a Constraint Programming model for a multi-manned assembly line balancing problem with walking workers and parallel stations. This model can be seen as an extension of multi-manned ALP with walking workers, and of RCPSP with parallel workstations. It provides an efficient way to evaluate how the number of workstations and

**Table 2** Production rates associated to each dataset and various layouts. Empty cells indicate that the 10 minutes timeout elapsed before a solution was found.

| | | Layout | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instance | Nb. tasks | $[1,1,1,1]$ | $[2,1,1]$ | $[1,2,1]$ | $[1,1,2]$ | $[3,1]$ | $[1,3]$ | $[4]$ |
| Instance 1 | 178 | 1260 | 1260 | 1260 | 1260 | 1024 | 900 | 855 |
| Instance 2 | 178 | 1260 | – | 1260 | 1260 | – | 954 | – |
| Instance 3 | 628 | 1200 | – | 900 | 1172 | – | 1160 | – |

their flow can help design a factory, by evaluating the production rates that it can attain. We detailed a constraint programming model in the OPL language that makes extensive use of interval variables, and validated the model by reproducing classical RCPSP benchmark results. We also used it to study assembly line implementations for industrial use cases.

There are several directions for pursuing this work. Better heuristics would probably improve the solver performance on the PSPLIB-based instances. Moreover, it would be interesting to address both workstation layout and resource sizing in a multi-objective approach. This would pave the way for the design of assembly lines with different configurations for low rate with low resource consumption, and high rates with high resource consumption.

## 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 Eduardo Álvarez-Miranda, Sebastián Chace, and Jordi Pereira. Assembly line balancing with parallel workstations. *International Journal of Production Research*, 59(21):6486–6506, 2021.

4 Felipe FB Araújo, Alysson M Costa, and Cristóbal Miralles. Two extensions for the alwabp: Parallel stations and collaborative approach. *International Journal of Production Economics*, 140(1):483–495, 2012.

5 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.

6 Gilles Audemard, Christophe Lecoutre, and Emmanuel Lonca. Proceedings of the 2023 XCSP3 competition. *CoRR*, abs/2312.05877, 2023. doi:10.48550/arXiv.2312.05877.

7 Zhongkai Bao, Lu Chen, and Kejun Qiu. An aircraft final assembly line balancing problem considering resource constraints and parallel task scheduling. *Computers & Industrial Engineering*, 182:109436, 2023.

8 Olga Battaïa and Alexandre Dolgui. Hybridizations in line balancing problems: A comprehensive review on new trends and formulations. *International Journal of Production Economics*, 250:108673, 2022.

9 Christian Becker and Armin Scholl. Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure. *European Journal of Operational Research*, 199(2):359–374, 2009.

10 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.

11 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.

**12** Joseph Bukchin and Jacob Rubinovitz. A weighted approach for assembly line design with station paralleling and equipment selection. *IIE transactions*, 35(1):73–85, 2003.

**13** Yin-Yann Chen, Chen-Yang Cheng, and Jia-Ying Li. Resource-constrained assembly line balancing problems with multi-manned workstations. *Journal of Manufacturing Systems*, 48:107–119, 2018.

**14** Zeynel Abidin Çil and Damla Kizilay. Constraint programming model for multi-manned assembly line balancing problem. *Computers & Operations Research*, 124:105069, 2020.

**15** Hongyan Ding, Cunbo Zhuang, and Jianhua Liu. Extensions of the resource-constrained project scheduling problem. *Automation in Construction*, 153:104958, 2023.

**16** Yunus Ege, Meral Azizoglu, and Nur E Ozdemirel. Assembly line balancing with station paralleling. *Computers & Industrial Engineering*, 57(4):1218–1225, 2009.

**17** Yagmur S Gök, Daniel Guimarans, Peter J Stuckey, Maurizio Tomasella, and Cemalettin Ozturk. Robust resource planning for aircraft ground operations. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21–24, 2020, Proceedings 17*, pages 222–238. Springer, 2020.

**18** Sönke Hartmann and Dirk Briskorn. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research*, 297(1):1–14, 2022.

**19** Rainer Kolisch and Sönke Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European journal of operational research*, 174(1):23–37, 2006.

**20** 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.

**21** Thiago Cantos Lopes, Adalberto Sato Michels, Celso Gustavo Stall Sikora, and Leandro Magatão. Balancing and cyclical scheduling of asynchronous mixed-model assembly lines with parallel stations. *Journal of Manufacturing Systems*, 50:193–200, 2019.

**22** Thiago Cantos Lopes, Giuliano Vidal Pastre, Adalberto Sato Michels, and Leandro Magatão. Flexible multi-manned assembly line balancing problem: Model, heuristic procedure, and lower bounds for line length minimization. *Omega*, 95:102063, 2020.

**23** Bahman Naderi, Ahmed Azab, and Katayoun Borooshan. A realistic multi-manned five-sided mixed-model assembly line balancing and scheduling problem with moving workers and limited workspace. *International Journal of Production Research*, 57(3):643–661, 2019.

**24** Cemalettin Öztürk, Semra Tunalı, Brahim Hnich, and Arslan Örnek. Balancing and scheduling of flexible mixed model assembly lines with parallel stations. *The International Journal of Advanced Manufacturing Technology*, 67:2577–2591, 2013.

**25** Cemalettin Öztürk, Semra Tunalı, Brahim Hnich, and Arslan Örnek. Cyclic scheduling of flexible mixed model assembly lines with parallel stations. *Journal of Manufacturing Systems*, 36:147–158, 2015.

**26** 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. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, pages 347–355, 2018.

**27** Stéphanie Roussel, Thomas Polacsek, and Anouck Chan. Assembly Line Preliminary Design Optimization for an Aircraft. In *CP 2023 (The 29th International Conference on Principles and Practice of Constraint Programming)*, Toronto, Canada, August 2023. doi:10.4230/LIPIcs.CP.2023.32.

**28** Stéphanie Roussel. Dataset for the Assembly Line Preliminary Design Optimization Problem, 2024. doi:10.57745/IQLQ7A.

29 Stéphanie Roussel and Xavier Pucel. Dataset for the Constraint Programming Model for Assembly Line Balancing and Scheduling with Walking Workers and Parallel Stations, 2024. doi:10.57745/EWXS9O.

30 Murat Şahin and Talip Kellegöz. Balancing multi-manned assembly lines with walking workers: problem definition, mathematical formulation, and an electromagnetic field optimisation algorithm. *International Journal of Production Research*, 57(20):6487–6505, 2019.

31 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. working paper or preprint, September 2021.

32 Celso Gustavo Stall Sikora, Thiago Cantos Lopes, and Leandro Magatão. Traveling worker assembly line (re) balancing problem: Model, reduction techniques, and real case studies. *European Journal of Operational Research*, 259(3):949–971, 2017.

33 Pascal Van Hentenryck, Laurent Michel, Laurent Perron, and J-C Régin. Constraint programming in opl. In *International Conference on Principles and Practice of Declarative Programming*, pages 98–116. Springer, 1999.


## A PSPLib Benchmarks Results

This appendix presents the results of execution of our solver on the first 150 instances of the PSPLib benchmark `j30rcp`. Table 3 is read as follows: Parameter value $p$ and Instance value $i$ describe the problem found in the file named `J30`$p$`_`$i$`.RCP` in the archive named `j30rcp.zip` found in the PSPlib [19] accessible at url `https://www.om-db.wi.tum.de/psplib/getdata_sm.html`. Each column in the "Cycle time" and "Computation time" groups corresponds to a different layout. A cycle time of $-1$ indicates that the solver failed to find a suitable schedule in the allocated time of 5 minutes. A computation time of 300 seconds (5 minutes) indicates that the solver used all its computation time. This means that the corresponding cycle time (if any) may be sub-optimal.

**Table 3** Detailed Results of the 150 first instances of the `j30rcp` benchmark for various number of stages and workstations.

| Parameter | Instance | Cycle time | | | | | | Computation time (seconds) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | [1] | [1,1] | [2] | [1,1,1] | [2,1] | [1,2] | [1] | [1,1] | [2] | [1,1,1] | [2,1] | [1,2] |
| 1 | 1 | 43 | 29 | 29 | 29 | 29 | 29 | 0.25 | 1.17 | 11.75 | 8.25 | 77.54 | 14.15 |
| 1 | 2 | 47 | 33 | 33 | 33 | 33 | 33 | 0.32 | 0.72 | 9.15 | 3.02 | 26.13 | 3.27 |
| 1 | 3 | 47 | 26 | 26 | 23 | 23 | 23 | 0.06 | 0.91 | 11.18 | 2.21 | 33.70 | 3.50 |
| 1 | 4 | 62 | 41 | 41 | 41 | 41 | 41 | 0.06 | 1.61 | 8.42 | 16.51 | 30.81 | 32.36 |
| 1 | 5 | 39 | 34 | 34 | 34 | 34 | 34 | 0.32 | 3.82 | 22.92 | 33.93 | 300 | 119.20 |
| 1 | 6 | 48 | 32 | 32 | 32 | 32 | 32 | 0.34 | 0.92 | 14.08 | 4.29 | 38.52 | 9.59 |
| 1 | 7 | 60 | 35 | 35 | 35 | 35 | 35 | 0.05 | 1.11 | 9.12 | 4.79 | 15.42 | 8.33 |
| 1 | 8 | 53 | 33 | 33 | 33 | 33 | 33 | 0.06 | 0.80 | 5.14 | 2.79 | 3.13 | 2.77 |
| 1 | 9 | 49 | 31 | 31 | 31 | 31 | 31 | 0.25 | 1.83 | 12.68 | 16.64 | 115.54 | 38.57 |
| 1 | 10 | 45 | 29 | 29 | 29 | 29 | 29 | 0.05 | 2.37 | 13.58 | 4.60 | 40.59 | 7.95 |
| 2 | 1 | 38 | 26 | 26 | 21 | 21 | 21 | 0.05 | 2.12 | 13.53 | 5.94 | 118.13 | 102.28 |
| 2 | 2 | 51 | 36 | 36 | 36 | 36 | 36 | 0.05 | 1.19 | 12.30 | 2.26 | 23.37 | 3.69 |
| 2 | 3 | 43 | 29 | 29 | 29 | 29 | 29 | 0.06 | 1.00 | 10.71 | 2.14 | 22.25 | 3.40 |
| 2 | 4 | 43 | 23 | 23 | 19 | 19 | 19 | 0.05 | 0.70 | 9.11 | 2.33 | 13.63 | 2.06 |
| 2 | 5 | 51 | 33 | 33 | 27 | 27 | 27 | 0.05 | 0.44 | 10.23 | 1.89 | 2.56 | 1.93 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 6 | 47 | 29 | 29 | 22 | 22 | 22 | 0.05 | 0.95 | 5.74 | 1.48 | 2.46 | 3.38 |
| 2 | 7 | 47 | 29 | 26 | 25 | 25 | 25 | 0.06 | 0.80 | 10.44 | 2.98 | 7.13 | 4.91 |
| 2 | 8 | 54 | 33 | 32 | 29 | 29 | 29 | 0.05 | 1.16 | 9.35 | 2.11 | 21.70 | 2.43 |
| 2 | 9 | 54 | 32 | 32 | 30 | 30 | 30 | 0.05 | 1.26 | 10.48 | 10.77 | 300 | 300 |
| 2 | 10 | 43 | 25 | 24 | 22 | 22 | 22 | 0.05 | 0.85 | 10.80 | 1.70 | 20.23 | 6.91 |
| 3 | 1 | 72 | 39 | 39 | 30 | 29 | 30 | 0.06 | 0.23 | 8.75 | 0.81 | 1.06 | 2.92 |
| 3 | 2 | 40 | 23 | 20 | 20 | 18 | 19 | 0.05 | 0.49 | 13.45 | 2.92 | 1.50 | 2.59 |
| 3 | 3 | 57 | 32 | 32 | 24 | 23 | 24 | 0.06 | 0.37 | 7.59 | 0.71 | 1.26 | 2.22 |
| 3 | 4 | 98 | 62 | 62 | 39 | 39 | 39 | 0.06 | 1.20 | 2.03 | 1.27 | 2.23 | 1.51 |
| 3 | 5 | 53 | 28 | 28 | 28 | 28 | 28 | 0.06 | 1.28 | 10.12 | 1.91 | 23.03 | 4.60 |
| 3 | 6 | 54 | 33 | 28 | 24 | 24 | 24 | 0.05 | 0.81 | 1.34 | 5.80 | 3.42 | 1.57 |
| 3 | 7 | 48 | 24 | 24 | 20 | 18 | 19 | 0.05 | 0.26 | 1.48 | 0.96 | 2.17 | 2.12 |
| 3 | 8 | 54 | 29 | 27 | 25 | 22 | 22 | 0.05 | 0.25 | 9.22 | 0.97 | 1.27 | 2.27 |
| 3 | 9 | 65 | 35 | 34 | 31 | 31 | 31 | 0.11 | 1.58 | 12.45 | 2.59 | 29.00 | 3.03 |
| 3 | 10 | 59 | 30 | 30 | 30 | 30 | 30 | 0.06 | 0.68 | 9.84 | 1.69 | 22.66 | 3.52 |
| 4 | 1 | 49 | 28 | 25 | 19 | 19 | 18 | 0.06 | 0.23 | 0.48 | 1.01 | 1.29 | 1.34 |
| 4 | 2 | 60 | 36 | 36 | 28 | 28 | 28 | 0.10 | 0.39 | 0.89 | 0.94 | 1.91 | 2.09 |
| 4 | 3 | 47 | 28 | 25 | 22 | 20 | 21 | 0.06 | 0.56 | 1.04 | 1.31 | 2.35 | 1.29 |
| 4 | 4 | 57 | 33 | 32 | 21 | 20 | 21 | 0.05 | 0.22 | 0.80 | 0.81 | 1.03 | 1.36 |
| 4 | 5 | 59 | 34 | 32 | 24 | 24 | 24 | 0.06 | 0.61 | 1.00 | 2.36 | 3.02 | 1.99 |
| 4 | 6 | 45 | 26 | 23 | 21 | 21 | 21 | 0.06 | 0.32 | 0.65 | 1.32 | 2.66 | 8.89 |
| 4 | 7 | 56 | 29 | 28 | 24 | 23 | 23 | 0.06 | 0.23 | 0.79 | 1.47 | 1.67 | 1.74 |
| 4 | 8 | 55 | 30 | 28 | 21 | 20 | 20 | 0.06 | 0.26 | 0.59 | 0.51 | 0.47 | 0.77 |
| 4 | 9 | 38 | 22 | 22 | 20 | 20 | 20 | 0.07 | 0.69 | 0.88 | 1.36 | 2.10 | 2.05 |
| 4 | 10 | 48 | 26 | 25 | 24 | 24 | 24 | 0.06 | 0.46 | 0.73 | 1.35 | 2.28 | 2.34 |
| 5 | 1 | 53 | 37 | 37 | 34 | 35 | 34 | 0.25 | 6.59 | 59.39 | 95.36 | 300 | 269.00 |
| 5 | 2 | 82 | 56 | 56 | 56 | 56 | 56 | 0.87 | 6.45 | 39.28 | 129.70 | 300 | 300 |
| 5 | 3 | 76 | 57 | 57 | 56 | 56 | 56 | 0.52 | 6.67 | 86.05 | 300 | 300 | 300 |
| 5 | 4 | 63 | 52 | 52 | 52 | 52 | 52 | 1.16 | 25.59 | 236.89 | 233.18 | 300 | 300 |
| 5 | 5 | 76 | 59 | 58 | 58 | 58 | 58 | 0.58 | 7.45 | 97.50 | 197.38 | 300 | 300 |
| 5 | 6 | 64 | 46 | 46 | 44 | 44 | 44 | 0.35 | 5.98 | 134.79 | 47.18 | 235.48 | 165.77 |
| 5 | 7 | 76 | 72 | 73 | 72 | 72 | 72 | 1.07 | 133.64 | 300 | 300 | 300 | 300 |
| 5 | 8 | 67 | 54 | 54 | 51 | 59 | 56 | 0.91 | 20.42 | 201.24 | 105.83 | 300 | 300 |
| 5 | 9 | 49 | 37 | 36 | 35 | 36 | 36 | 0.37 | 7.11 | 75.86 | 184.16 | 300 | 300 |
| 5 | 10 | 70 | 55 | 54 | 52 | 53 | 53 | 0.63 | 11.10 | 99.64 | 126.15 | 300 | 300 |
| 6 | 1 | 59 | 42 | 42 | 42 | 42 | 42 | 0.29 | 11.41 | 95.94 | 300 | 300 | 300 |
| 6 | 2 | 51 | 36 | 36 | 35 | 34 | 34 | 0.13 | 1.85 | 13.04 | 26.98 | 283.05 | 74.27 |
| 6 | 3 | 48 | 31 | 31 | 30 | 30 | 30 | 0.08 | 1.83 | 15.05 | 78.84 | 300 | 300 |
| 6 | 4 | 42 | 33 | 33 | 32 | 33 | 32 | 0.50 | 11.12 | 300 | 300 | 300 | 300 |
| 6 | 5 | 67 | 51 | 51 | 48 | 50 | 48 | 0.26 | 7.67 | 74.17 | 93.55 | 300 | 300 |
| 6 | 6 | 37 | 26 | 25 | 24 | 24 | 24 | 0.05 | 3.08 | 17.31 | 21.34 | 300 | 251.14 |
| 6 | 7 | 46 | 30 | 30 | 29 | 30 | 29 | 0.05 | 1.95 | 15.39 | 16.08 | 300 | 227.00 |
| 6 | 8 | 39 | 30 | 30 | 30 | 30 | 31 | 0.05 | 3.78 | 123.51 | 300 | 300 | 300 |
| 6 | 9 | 51 | 35 | 35 | 35 | 35 | 35 | 0.06 | 1.61 | 8.64 | 7.15 | 60.12 | 10.62 |
| 6 | 10 | 61 | 44 | 43 | 43 | 43 | 45 | 0.37 | 16.67 | 201.23 | 300 | 300 | 300 |
| 7 | 1 | 55 | 29 | 28 | 25 | 25 | 25 | 0.06 | 1.63 | 11.28 | 2.06 | 6.03 | 4.25 |
| 7 | 2 | 42 | 28 | 28 | 27 | 27 | 27 | 0.05 | 1.86 | 12.51 | 6.13 | 11.71 | 20.75 |
| 7 | 3 | 42 | 28 | 27 | 26 | 26 | 26 | 0.06 | 2.66 | 6.12 | 4.49 | 61.24 | 11.89 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 4 | 44 | 30 | 29 | 25 | 25 | 25 | 0.05 | 1.83 | 12.79 | 8.00 | 72.16 | 33.25 |
| 7 | 5 | 44 | 31 | 30 | 30 | 30 | 30 | 0.09 | 3.07 | 40.28 | 144.03 | 300 | 300 |
| 7 | 6 | 35 | 22 | 20 | 19 | 18 | 18 | 0.06 | 0.92 | 14.24 | 4.04 | 12.15 | 23.66 |
| 7 | 7 | 50 | 33 | 32 | 30 | 29 | 30 | 0.06 | 1.80 | 11.25 | 5.07 | 16.05 | 19.83 |
| 7 | 8 | 44 | 35 | 34 | 33 | 34 | 33 | 0.06 | 2.10 | 16.16 | 26.62 | 300 | 155.17 |
| 7 | 9 | 60 | 33 | 33 | 31 | 31 | 30 | 0.05 | 1.02 | 13.21 | 2.76 | 22.47 | 3.02 |
| 7 | 10 | 49 | 33 | 31 | 29 | 29 | 29 | 0.26 | 1.52 | 11.14 | 3.23 | 30.24 | 14.02 |
| 8 | 1 | 44 | 26 | 25 | 23 | 23 | 23 | 0.06 | 1.15 | 1.03 | 2.40 | 9.16 | 12.08 |
| 8 | 2 | 51 | 30 | 26 | 21 | 21 | 21 | 0.07 | 0.21 | 0.64 | 1.49 | 3.37 | 2.64 |
| 8 | 3 | 53 | 29 | 27 | 25 | 25 | 25 | 0.05 | 0.56 | 1.07 | 1.27 | 2.59 | 2.74 |
| 8 | 4 | 48 | 26 | 24 | 22 | 21 | 21 | 0.06 | 0.43 | 0.98 | 1.33 | 2.06 | 3.00 |
| 8 | 5 | 58 | 32 | 32 | 30 | 30 | 30 | 0.06 | 1.03 | 1.05 | 4.50 | 7.36 | 17.72 |
| 8 | 6 | 47 | 27 | 26 | 25 | 24 | 24 | 0.06 | 0.83 | 1.21 | 21.96 | 49.80 | 51.37 |
| 8 | 7 | 41 | 23 | 21 | 18 | 18 | 18 | 0.07 | 0.36 | 0.08 | 1.36 | 3.93 | 3.60 |
| 8 | 8 | 51 | 30 | 28 | 25 | 25 | 25 | 0.06 | 1.53 | 1.49 | 3.28 | 6.08 | 6.51 |
| 8 | 9 | 39 | 22 | 20 | 19 | 19 | 19 | 0.16 | 0.54 | 0.77 | 5.67 | 173.10 | 74.25 |
| 8 | 10 | 67 | 36 | 34 | 25 | 25 | 25 | 0.06 | 0.09 | 0.29 | 1.24 | 3.89 | 2.06 |
| 9 | 1 | 83 | 75 | -1 | 75 | 80 | 75 | 3.57 | 300 | 300 | 300 | 300 | 300 |
| 9 | 2 | 92 | 91 | -1 | 89 | 91 | 89 | 50.50 | 300 | 300 | 300 | 300 | 300 |
| 9 | 3 | 68 | 64 | 60 | 56 | 60 | 59 | 1.18 | 300 | 300 | 300 | 300 | 300 |
| 9 | 4 | 71 | 63 | 64 | 62 | 66 | 69 | 1.48 | 128.39 | 300 | 300 | 300 | 300 |
| 9 | 5 | 70 | 58 | 58 | 57 | 63 | 61 | 0.75 | 32.52 | 191.66 | 300 | 300 | 300 |
| 9 | 6 | 59 | 48 | -1 | 51 | 50 | 50 | 1.16 | 211.58 | 300 | 300 | 300 | 300 |
| 9 | 7 | 63 | 52 | -1 | 53 | 55 | 56 | 1.74 | 198.00 | 300 | 300 | 300 | 300 |
| 9 | 8 | 91 | 79 | -1 | 79 | 82 | 81 | 1.46 | 300 | 300 | 300 | 300 | 300 |
| 9 | 9 | 63 | 52 | 52 | 52 | 61 | 52 | 2.58 | 300 | 300 | 300 | 300 | 300 |
| 9 | 10 | 88 | 79 | 80 | 76 | 76 | 80 | 3.53 | 300 | 300 | 300 | 300 | 300 |
| 10 | 1 | 42 | 31 | 31 | 30 | 31 | 31 | 0.06 | 6.13 | 47.72 | 208.58 | 300 | 300 |
| 10 | 2 | 56 | 43 | -1 | 43 | 44 | 44 | 0.42 | 35.87 | 300 | 300 | 300 | 300 |
| 10 | 3 | 62 | 48 | 51 | 49 | 48 | 49 | 0.38 | 213.37 | 300 | 300 | 300 | 300 |
| 10 | 4 | 58 | 44 | 44 | 44 | 45 | 44 | 0.28 | 17.68 | 300 | 300 | 300 | 300 |
| 10 | 5 | 41 | 35 | 34 | 34 | 35 | 34 | 0.11 | 300 | 300 | 300 | 300 | 300 |
| 10 | 6 | 44 | 34 | 34 | 34 | 35 | 34 | 0.36 | 28.65 | 300 | 300 | 300 | 300 |
| 10 | 7 | 49 | 34 | 34 | 33 | 34 | 33 | 0.07 | 41.67 | 300 | 300 | 300 | 300 |
| 10 | 8 | 54 | 41 | -1 | 40 | 41 | 42 | 0.38 | 45.85 | 300 | 300 | 300 | 300 |
| 10 | 9 | 49 | 31 | 32 | 31 | 31 | 31 | 0.05 | 26.49 | 300 | 300 | 300 | 300 |
| 10 | 10 | 41 | 31 | 31 | 30 | 31 | 31 | 0.31 | 15.29 | 300 | 300 | 300 | 300 |
| 11 | 1 | 54 | 44 | 44 | 44 | 44 | 44 | 0.08 | 51.93 | 300 | 300 | 300 | 300 |
| 11 | 2 | 56 | 43 | 42 | 42 | 43 | 42 | 0.13 | 10.68 | 163.90 | 300 | 300 | 300 |
| 11 | 3 | 81 | 43 | 41 | 37 | 38 | 38 | 0.06 | 0.95 | 15.16 | 142.67 | 300 | 300 |
| 11 | 4 | 63 | 42 | 41 | 40 | 41 | 41 | 0.05 | 4.83 | 55.68 | 300 | 300 | 300 |
| 11 | 5 | 49 | 40 | 40 | 40 | 40 | 40 | 0.34 | 79.96 | 300 | 300 | 300 | 300 |
| 11 | 6 | 44 | 30 | 30 | 29 | 30 | 29 | 0.06 | 14.26 | 246.49 | 300 | 300 | 300 |
| 11 | 7 | 36 | 27 | 26 | 26 | 27 | 27 | 0.06 | 10.49 | 300 | 300 | 300 | 300 |
| 11 | 8 | 62 | 44 | 43 | 44 | 43 | 45 | 0.07 | 17.88 | 281.82 | 300 | 300 | 300 |
| 11 | 9 | 67 | 41 | 41 | 40 | 41 | 41 | 0.06 | 3.70 | 27.53 | 300 | 300 | 300 |
| 11 | 10 | 38 | 27 | 27 | 26 | 26 | 27 | 0.06 | 3.00 | 19.09 | 174.05 | 300 | 300 |
| 12 | 1 | 47 | 29 | 28 | 26 | 26 | 26 | 0.15 | 1.76 | 2.51 | 73.81 | 300 | 300 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 2 | 46 | 30 | 30 | 30 | 30 | 30 | 0.06 | 2.15 | 45.26 | 300 | 300 | 300 |
| 12 | 3 | 37 | 23 | 23 | 22 | 22 | 22 | 0.06 | 2.17 | 14.73 | 74.48 | 300 | 300 |
| 12 | 4 | 63 | 35 | 32 | 29 | 29 | 29 | 0.08 | 0.29 | 1.31 | 50.03 | 300 | 300 |
| 12 | 5 | 47 | 24 | 24 | 21 | 21 | 21 | 0.07 | 0.29 | 0.89 | 35.54 | 300 | 300 |
| 12 | 6 | 53 | 31 | 31 | 29 | 30 | 29 | 0.06 | 2.76 | 10.51 | 264.60 | 300 | 300 |
| 12 | 7 | 55 | 30 | 28 | 27 | 27 | 27 | 0.07 | 0.75 | 2.29 | 300 | 300 | 300 |
| 12 | 8 | 35 | 19 | 18 | 18 | 18 | 18 | 0.07 | 0.77 | 0.72 | 5.97 | 300 | 300 |
| 12 | 9 | 52 | 30 | 29 | 28 | 28 | 29 | 0.06 | 1.66 | 8.45 | 300 | 300 | 300 |
| 12 | 10 | 57 | 32 | 29 | 26 | 26 | 26 | 0.07 | 0.24 | 0.95 | 60.81 | 300 | 300 |
| 13 | 1 | 58 | 55 | -1 | 56 | 56 | 55 | 12.99 | 300 | 300 | 300 | 300 | 300 |
| 13 | 2 | 62 | 61 | -1 | 61 | 61 | 60 | 121.27 | 300 | 300 | 300 | 300 | 300 |
| 13 | 3 | 76 | 73 | -1 | 73 | 73 | 73 | 16.92 | 300 | 300 | 300 | 300 | 300 |
| 13 | 4 | 72 | 64 | -1 | 64 | 67 | 64 | 4.85 | 300 | 300 | 300 | 300 | 300 |
| 13 | 5 | 67 | 65 | -1 | 65 | 65 | 65 | 30.07 | 300 | 300 | 300 | 300 | 300 |
| 13 | 6 | 64 | 60 | 64 | 60 | 61 | 60 | 34.34 | 300 | 300 | 300 | 300 | 300 |
| 13 | 7 | 77 | 76 | -1 | 76 | -1 | 75 | 13.47 | 300 | 300 | 300 | 300 | 300 |
| 13 | 8 | 106 | 102 | -1 | 102 | 109 | 97 | 47.01 | 300 | 300 | 300 | 300 | 300 |
| 13 | 9 | 71 | 65 | -1 | 65 | 69 | 67 | 1.85 | 300 | 300 | 300 | 300 | 300 |
| 13 | 10 | 64 | 56 | 56 | 56 | 55 | 56 | 3.98 | 300 | 300 | 300 | 300 | 300 |
| 14 | 1 | 50 | 40 | 41 | 41 | 40 | 40 | 0.60 | 236.77 | 300 | 300 | 300 | 300 |
| 14 | 2 | 53 | 49 | -1 | 49 | 49 | 49 | 1.03 | 300 | 300 | 300 | 300 | 300 |
| 14 | 3 | 58 | 52 | 52 | 50 | 51 | 52 | 0.36 | 300 | 300 | 300 | 300 | 300 |
| 14 | 4 | 50 | 42 | 42 | 41 | 43 | 41 | 0.64 | 300 | 300 | 300 | 300 | 300 |
| 14 | 5 | 52 | 36 | 37 | 37 | 37 | 36 | 0.10 | 121.81 | 300 | 300 | 300 | 300 |
| 14 | 6 | 35 | 30 | 30 | 29 | 29 | 29 | 0.07 | 300 | 300 | 300 | 300 | 300 |
| 14 | 7 | 50 | 46 | 45 | 44 | 44 | 46 | 1.01 | 300 | 300 | 300 | 300 | 300 |
| 14 | 8 | 54 | 42 | 42 | 42 | 41 | 42 | 0.06 | 300 | 300 | 300 | 300 | 300 |
| 14 | 9 | 46 | 39 | 40 | 39 | 40 | 41 | 0.83 | 258.98 | 300 | 300 | 300 | 300 |
| 14 | 10 | 61 | 43 | 44 | 43 | 44 | 43 | 0.29 | 10.91 | 300 | 300 | 300 | 300 |
| 15 | 1 | 46 | 34 | 34 | 35 | 34 | 35 | 0.05 | 300 | 300 | 300 | 300 | 300 |
| 15 | 2 | 47 | 29 | 30 | 29 | 30 | 29 | 0.06 | 21.71 | 300 | 300 | 300 | 300 |
| 15 | 3 | 48 | 34 | 34 | 35 | 34 | 34 | 0.05 | 300 | 300 | 300 | 300 | 300 |
| 15 | 4 | 48 | 27 | 24 | 24 | 24 | 24 | 0.06 | 0.26 | 7.58 | 300 | 300 | 300 |
| 15 | 5 | 58 | 53 | 52 | 52 | 54 | 53 | 0.94 | 300 | 300 | 300 | 300 | 300 |
| 15 | 6 | 67 | 45 | 46 | 46 | 46 | 46 | 0.14 | 198.44 | 300 | 300 | 300 | 300 |
| 15 | 7 | 47 | 33 | 33 | 33 | 34 | 33 | 0.05 | 23.22 | 300 | 300 | 300 | 300 |
| 15 | 8 | 50 | 39 | 39 | 39 | 39 | 39 | 0.06 | 300 | 300 | 300 | 300 | 300 |
| 15 | 9 | 54 | 35 | 36 | 36 | 36 | 35 | 0.06 | 300 | 300 | 300 | 300 | 300 |
| 15 | 10 | 65 | 40 | 40 | 40 | 40 | 40 | 0.06 | 6.01 | 112.92 | 300 | 300 | 300 |