Modeling and Explaining an Industrial Workforce Allocation Problem (Extended Abstract*)

Ignace Bleukx[1] , **Ryma Boumazouza**[2,3] , **Tias Guns**[1] , **Nadine Laage**[4] and **Guillaume Poveda**[2]
[1]KU Leuven, Department of Computer Science
[2]Airbus SAS
[3]Institut de Recherche Technologique Saint-Exupery
[4]Airbus Aerostructures GmbH
{ignace.bleukx, tias.guns}@kuleuven.be
{ryma.boumazouza, nadine.laage, guillaume.poveda}@airbus.com

Abstract

We present an industrial case on workforce allocation and scheduling in the aircraft manufacturing industry, where available teams need to be assigned to logistical operations. This application presents several challenges, such as the scale of the problem, the need for fair workload distribution, and the need for methods to mitigate unforeseen disruptions due to technical malfunctions or incompatible weather conditions. We compare different Constraint Programming (CP) models for the allocation and scheduling problems, with extra focus on modeling the workload balancing component. Additionally, we investigate automatic rescheduling methods for restoring feasibility after a disruption invalidates the precomputed schedule. Our results show that by using appropriate modeling techniques, the problem can be solved in reasonable time, thereby producing fair schedules. Additionally, we show how invalidated schedules can be restored efficiently to help human operators in resolving disruptions to the schedule.

1 Introduction

The assignment of workers to different tasks is a recurring challenge in many industries. Effective allocation of workers directly impacts key business metrics such as productivity, cost management, and overall organizational performance. Solving an industrial workforce allocation problem requires considering several factors such as such as employee availability, required qualifications of workers for a task, regulatory requirements, and task priorities.

We address the problem of workforce allocation for logistical tasks among worker teams at an aircraft manufacturing site, where hundreds of tasks need to be allocated on a daily basis. These logistical tasks are interleaved with production activities on assembly lines [Boysen *et al.*, 2022], and need to be aligned with timings of higher-level logistics involving deliveries of raw materials or components provided by external parties via trucks, ships, or cargo aircraft. Delivered components then have to be transported to internal hangars at the right time, where they will be assembled with other aircraft parts. We incorporate these operational requirements into the allocation and scheduling problem as part of the input data. More specifically, each task has a predefined release time, duration, and deadline, and certain tasks share precedences. E.g., when two parts must be delivered in sequence before assembly. Apart from explicit precedence constraints, there are no requirements on the task ordering, as the duration of any task includes returning to the home base of the manufacturing plant. Hence, the transition time between the end location of one task and the starting location of the next is included in the duration of each task. This differs from the traditional definition of a workforce-scheduling and routing problem (WSRP) [Castillo-Salazar *et al.*, 2016] and allows to formulate the problem as a pure scheduling problem instead.

The goal is to find an allocation of tasks to working teams, such that the total number of required teams is minimized. Additionally, the workload should be distributed fairly among the teams. That is, the allocation of tasks should be done such that all teams do a similar amount of work.

Currently, transportation tasks are manually allocated and scheduled by a team of planners. These planners also coordinate the fine-grained synchronization among teams to ensure dependencies between tasks are satisfied. During the working day, they continuously monitor real-time updates and adjust schedules as needed to handle unforeseen disruptions. Such disruptions can arise for multiple reasons, including:

- **Weather-related:** Extreme wind or rain prevents the use of some hangars.
- **External logistics:** Delays or cancellations from suppliers impact the arrival time of essential parts.
- **Human resource:** Unexpected worker absences or team meetings temporarily reduce workforce availability.
- **Material availability:** Vehicles like trucks or trailers may become temporarily unavailable due to technical breakdowns, accidents or maintenance operations.

Large disruptions can impact the predefined plan in such a way that the set of disrupted tasks cannot be reallocated

*The original paper won the Best Application Paper award at the CP'25 conference [Bleukx et al., 2025]

to other teams. This requires the planners to revise the plan, which may involve multiple re-allocation and/or re-scheduling operations. As this is tedious and time-consuming to do by hand, planners often drop disrupted tasks from the current planning horizon and move them to the next one. To help the planners revise the allocation more efficiently, we propose to automatically repair disrupted schedules and show several alternatives to the planners [Senthooran *et al.*, 2023]. This allows them to further adapt or merge alternatives based on their experience and knowledge about the problem.

2 A CP-model for Workforce Allocation

We propose to solve this complex scheduling and allocation problem using Constraint Programming. The model needs to solve two aspects of the problem: (1) allocate each task to a compatible team, and (2) schedule all tasks such that those tasks assigned to the same team do not overlap. I.e., such that each team executes at most one task at the same time. For objectives, we require a minimal number of worker teams in the schedule and a fair workload distribution. In what follows, we discuss the input data, the mathematical formulation of the constraints and objectives, and one advanced modeling technique to improve the runtime of the solver.

Input data

- $\mathcal{T}$: a set of n tasks to assign to teams.
- $\mathcal{W}$: a set of m worker-teams to assign tasks to.
- $Comp \in \mathbb{B}^{n \times m}$ where $Comp_{t,w}$ indicates team w and task t are compatible.
- D: the fixed durations of each task, with $d_i \in D$ the duration of task i.
- $\mathcal{G}$: set of task clusters where tasks in cluster $g \subseteq \mathcal{T}$ should be executed by the same team.
- $\mathcal{R}$: set with the release time of each task.
- $\mathcal{E}$: set with the deadline of each task.
- $Prec$: a set of precedence tuples (t, t').
- Cal_w: a set of tuples (c_s, c_d) that represent intervals during which team w is off-duty.

Decision variables and constraints Our model is inspired by flexible-jobshop problems [Beck and Fox, 2000], where a set of *jobs* (tasks) need to be scheduled on a compatible *resource* (a worker team). To indicate which team a task is assigned to, we use a $n \times m$ Boolean matrix with variables $x_{t,w}$ indicating task t is assigned to team w. To model the scheduling aspect of the problem, we rely on scheduling-specific *global constraints* developed by the CP community [van Hoeve and Katriel, 2006]. These global constraints reason over uninterruptible tasks, which are represented as *interval variables* $\textsc{Interval}(s, d)$ with an integer variable start s and an integer variable d as duration. An extension to interval variables is *optional intervals* $\textsc{OptInterval}(s, d, p)$ where Boolean variable p indicates whether the task is *present* or *absent* in the schedule [Laborie *et al.*, 2009]. In our model, we duplicate each task for each team using an optional interval, and determine whether the interval is present in the solution using the variable $x_{t,w}$.

Each task is allocated to a team:

$$\forall t \in \mathcal{T} : \sum\nolimits_{w \in \mathcal{W}} x_{t,w} = 1$$

Tasks are only assigned to compatible teams:

$$\forall t \in \mathcal{T}, \forall w \in \mathcal{W} : \neg Comp_{t,w} \Rightarrow \neg x_{t,w}$$

All tasks in cluster g are assigned to the same team:

$$\forall g \in \mathcal{G}, \forall w \in \mathcal{W} : \textsc{AllEqual}(\{x_{t,w} \mid t \in g\})$$

Tasks are scheduled within their allowed time frame:

$$\forall t \in \mathcal{T} : s_t \geq r_t \wedge s_t + d_t \leq e_t$$

Enforce precedences between tasks:

$$\forall (t, t') \in Prec : s_t + d_t \leq s_{t'}$$

Teams cannot perform overlapping tasks and can only work when available.

$$\textsc{NoOverlap}(\{\textsc{OptInterval}(s_t, d_t, x_{t,w}) \mid t \in \mathcal{T}\} \cup \{\textsc{Interval}(c_s, c_d) \mid (c_s, c_d) \in Cal_w\})$$

Auxiliary variables and objectives We introduce auxiliary variables $used_w \in \mathbb{B}^m$ that indicate whether team w is used, and integer variables $time_w \in \mathbb{N}^m$ that indicate the time worked by each team. These auxiliary variables relate to the decision variables x as follows:

A team is used if a task is allocated to it:

$$\forall w \in \mathcal{W} : \bigvee\nolimits_{t \in \mathcal{T}} x_{t,w} \Rightarrow used_w \tag{1}$$

Definition of the time worked by a team, only enforced if the team is used:

$$\forall w \in \mathcal{W} : used_w \Rightarrow \sum\nolimits_{t \in \mathcal{T}} d_t \cdot x_{t,w} = time_w$$

We consider two objectives in the problem: the number of teams used and the *fairness* of the schedule, formulated as *time dispersion*, which is the difference between the maximum and minimum workload of any pair of teams. This corresponds to a *spread-based* fairness measure from the literature [Chen and Hooker, 2021; Dönmez *et al.*, 2023]. Formally, our model becomes a bi-objective lexicographic minimization problem with primary objective $\sum_{w \in \mathcal{W}} used_w$ and secondary objective $\max_{w \in \mathcal{W}} time_w - \min_{w \in \mathcal{W}} time_w$.

Redundant constraints During experimentation, we found that when minimizing the number of required teams, the solver is often stuck on proving optimality of the solution. That is, a solution was found with an *optimal objective value*, but the solver could not determine a better solution cannot exist. We want to help the solver by enforcing a reasonable lower-bound on the number of teams. This is done by adding a *redundant* or *implied* constraint to the problem, which is common in CP [Smith, 2006]. For our setting, we add the following CUMULATIVE constraint to the problem.

$$\textsc{Cumulative}(\{\textsc{Interval}(s_t, d_t) \mid t \in \mathcal{T}\}, \\ demand = 1, \\ capacity = \sum\nolimits_{w \in \mathcal{W}} used_w)$$

This constraint effectively treats each team as equivalent and ignores any team-task compatibility or calendar restrictions. This allows to model the set of worker teams as a *pooled* resource, with a capacity equal to the number of teams that are actually working. Clearly, any solution to the original model also satisfies this constraint, as it is relaxation of the original model. Efficient propagators for the CUMULATIVE constraint can efficiently compute lower bounds on the capacity of this resource [Schutt *et al.*, 2011], which for our problem means the solver can efficiently derive a lower-bound on the objective of the optimization problem.

3 Explaining and Resolving Disruptions

We now outline our efforts to develop an explainable decision-making tool for the allocation-and-scheduling problem, with the potential for generalization to similar use cases. Our primary application of explanation techniques will be to handle disruptions, which invalidate the precomputed schedule and require the planner to re-allocate and/or re-schedule a subset of the tasks. We here focus on *resolving* disruptions in the schedule using automatic re-allocation and shifting of tasks, while more elaborate explanation techniques are discussed in the full version of our paper [Bleukx *et al.*, 2025].

3.1 Disruption scenarios

From the planner's perspective, the scenario is as follows: an initial schedule is established for the upcoming shift, with an estimate of available teams. This schedule can be computed using the CP model from the previous section or it can be manually constructed by the planners. As discussed previously, various disruptions can affect the plan and possibly make the foreseen allocation infeasible. In response, the planner must adapt and reallocate the tasks to available resources such that as many tasks as possible can still be executed, without wasting resources. To do so, the planners will try to reallocate tasks to other, free teams. However, in practice, this is often not possible as task-compatible teams are usually already occupied with other tasks. Instead, some tasks need to be delayed in order to free up a worker team in the required time window. However, this process is tedious and time-consuming for planners to do by hand; as all operational constraints must remain satisfied during the rescheduling process. Instead, we propose to automatically revise the schedule, as outlined in the next section.

3.2 Automated rescheduling

To restore the feasibility of the schedule after a disruption, we can, in theory, simply rerun the solver with the changed input data. E.g., by changing a team's availability for a certain time window during which the duration happened. However, the resulting solution may be significantly different from the current schedule, and several teams and tasks will be impacted by the revised schedule. Inspired by classic feasibility restoration and re-planning approaches [Fox *et al.*, 2006; Senthooran *et al.*, 2023], we instead propose to re-optimize the schedule, while staying close to the one that was originally constructed. More specifically, based on the original assignment α, we define several *similarity* objectives. These objectives ensure the new solution is as *similar* as possible to the original assignment. We define five such similarity objectives, which minimize the number of re-allocated tasks, the number of delayed tasks, and the magnitude of delays.

Maximize the number of tasks that are allocated:

$$maximize \quad (\sum_{t \in \mathcal{T}}(\sum_{w \in \mathcal{W}} x_{t,w})) = (\sum_{w \in \mathcal{W}} \alpha(x_{t,w})) \quad (2)$$

Minimize the number of tasks that are re-allocated:

$$minimize \quad \sum_{w \in \mathcal{W}} \sum_{t \in \mathcal{T}} (\alpha(x_{t,w}) \neq x_{t,w}) \quad (3)$$

Minimize the number of shifted tasks:

$$minimize \quad \sum_{t \in \mathcal{T}} (\alpha(s_t) \neq s_t) \quad (4)$$

Minimize the total shift in time of the tasks:

$$minimize \quad \sum_{t \in \mathcal{T}} |\alpha(s_t) - s_t| \quad (5)$$

Minimize the maximum shift in time of the tasks:

$$minimize \quad \max_{t \in \mathcal{T}} |\alpha(s_t) - s_t| \quad (6)$$

Instead of optimizing only one of these similarity objectives, we combine them into a Lexicographic Multi-Objective Optimization Problem (LMOCOP) [Schaus and Hartert, 2013; Cortes *et al.*, 2023]. This allows us to adjust the ordering of objectives, which influences the final repaired schedules after disruptions. Figure 1 shows two repairs for different orders of the sub-objectives on the same instance. Depending on the ordering of the objectives, a different trade-off between reallocating and rescheduling tasks is found.

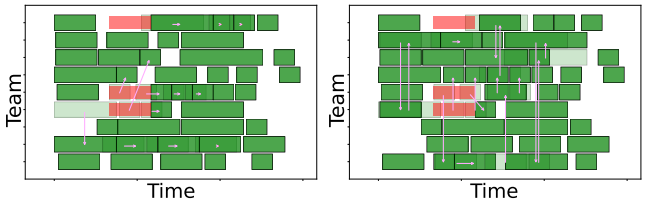

Figure 1: Different repaired schedules after disruptions, prioritizing minimal reallocations (left) over minimal nb of shifted tasks (right).

By optimizing this multi-objective optimization problem for each order, we can present the decision maker with multiple alternatives on how to resolve the disruption. They can then pick one or combine alternatives to construct a final schedule.

Optimization order	#not-done	#realloc	#shift	sum-shift	max-shift
Reallocation first (A)	1.00	**8.37**	23.21	289.21	24.04
Shift first (B)	0.96	22.60	17.59	244.75	21.40
Sum Shift first (C)	0.98	22.48	**16.66**	**198.93**	16.15
Max Shift first (D)	0.98	22.95	17.54	199.71	**14.91**

Table 1: Average objective value for feasibility restoration, grouped per lexicographic ordering of the similarity objectives.

4 Experimental Evaluation

In this section, we investigate the runtime performance of CP solvers for solving the large-scale workforce scheduling and allocation problem, and we evaluate our approach for automatically resolving disruptions. All code and benchmarks are available on github [1].

4.1 Solving the Workforce Allocation Problem

We first compare the runtime of the OR-Tools when considering a single objective: minimization of the number of teams. The left-hand side of Figure 2 shows a cactus plot, counting the number of solved instances within a given runtime. It is clear that the addition of the redundant cumulative constraint is essential for the solver to prove optimality of the solution. Indeed, the solver was able to solve 253 instances to optimality using the redundant constraint, compared to only 117 without.

Next, we compare the objective value over time found by the solver on the right-hand side of the figure. While optimal solutions were never found for our benchmark when considering both objectives, the figure clearly shows the solver finds a solution with a *good* objective value early on in the search. Indeed, after 10 seconds of solve-time, the time dispersion drops to 6.2 minutes on average.

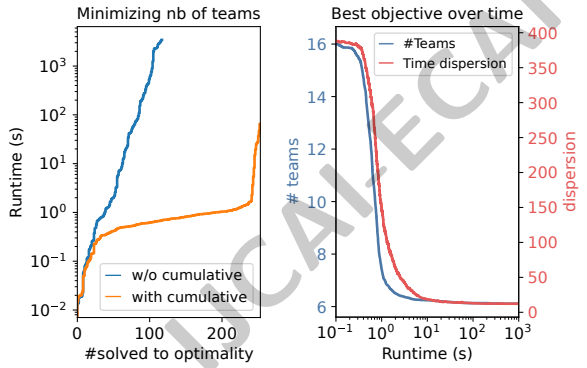

Figure 2: Experimental results for solving the allocation and scheduling problem, showing the number of instances solved to optimality when minimizing for the number of teams (left), and the best objective found over time when considering both objectives (right)

4.2 Rescheduling after disruptions

We now investigate our method for restoring feasibility by computing fine-grained alterations to the schedule as described in Section 3.2. To evaluate our method in a broad range of disruption scenarios, we used a generator with adjustable parameters for simulating disruptions. These generator parameters were designed in coordination with the planning team to cover a range of realistic and complex scenarios that arise in their daily practice. We solve the lexicographic optimization problem by iteratively optimizing for each objective. For each solver call (one for each objective), we set a time limit of 5 seconds, such that the total time for feasibility restoration is limited to 25 seconds.

Table 1 shows that by altering the order of the objectives, the resulting repaired schedule can be influenced significantly. As expected, there is a clear trade-off between shifting tasks and re-allocating tasks after a disruption. In "Reallocation first (A)", which prioritizes minimization of re-allocated tasks, 8.37 tasks are re-allocated on average, while for the other orders between 22 and 23 tasks are re-allocated. Conversely, prioritizing minimization of shift-related metrics amounts to a lower number of shifted tasks (16-17) at the cost of more re-allocations.

Interestingly, optimizing the total magnitude of shifted tasks, results in less tasks that are shifted on average, compared to minimizing `#shift` directly. Further investigation reveals that this is due to `#shift` being a more difficult objective to optimize compared to `sum-shift`, thereby often reaching the time limit of 5s. Indeed, in only 60% of the instances, OR-Tools was able to provide an optimal solution when optimizing the `#shift` similarity objective first. In "Sum Shift First (C)", the `sum-shift` similarity objective was proven optimal in 80% of the instances and, by minimizing this simpler objective first, the resulting repaired schedule contains fewer shifted tasks overall.

5 Conclusion and Future Work

We have presented a workforce-allocation and scheduling problem from the aircraft manufacturing industry. We have shown that, by using the right modeling techniques, CP solvers are able to find good solutions for this problem. In the future, we plan to adapt our model, allowing for direct transfers between locations, which requires extending the formulation to a full allocation-and-routing problem [Castillo-Salazar *et al.*, 2016]. Additionally, we have identified several disruption scenarios and investigated how to support a decision maker in resolving them efficiently. We deem it feasible to deploy such an interactive system as a decision support system for the planners. A future user study will enable us to further tune the type of explanations generated by the system, and, going further, could allow us to learn preferences of the planners in repairing the schedule [Guerreiro *et al.*, 2023; Benabbou and Lust, 2019; Toffano *et al.*, 2022].

[1] github.com/ML-KULeuven/Explainable-Workforce-Scheduling

Acknowledgments

This work was partially supported by the European Research and Innovation program TUPLES (101070149). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[Beck and Fox, 2000] J Christopher Beck and Mark S Fox. Constraint-directed techniques for scheduling alternative activities. *Artificial Intelligence*, 121(1-2):211–250, 2000.

[Benabbou and Lust, 2019] Nawal Benabbou and Thibaut Lust. An interactive polyhedral approach for multi-objective combinatorial optimization with incomplete preference information. In *International Conference on Scalable Uncertainty Management*, pages 221–235. Springer, 2019.

[Bleukx *et al.*, 2025] Ignace Bleukx, Ryma Boumazouza, Tias Guns, Nadine Laage, and Guillaume Poveda. Modeling and explaining an industrial workforce allocation and scheduling problem. In *CP*, volume 340 of *LIPIcs*, pages 6:1–6:24. Schloss Dagstuhl - Leibniz-Zentrum fur Informatik, 2025.

[Boysen *et al.*, 2022] 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.

[Castillo-Salazar *et al.*, 2016] J Arturo Castillo-Salazar, Dario Landa-Silva, and Rong Qu. Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, 239:39–67, 2016.

[Chen and Hooker, 2021] Violet Xinying Chen and JN Hooker. A guide to formulating equity and fairness in an optimization model. *Preprint*, pages 162–174, 2021.

[Cortes *et al.*, 2023] João Cortes, Inês Lynce, and Vasco Manquinho. New core-guided and hitting set algorithms for multi-objective combinatorial optimization. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part II*, volume 13994 of *Lecture Notes in Computer Science*, pages 55–73. Springer, 2023.

[Dönmez *et al.*, 2023] Zehranaz Dönmez, M Ayyildiz, Benay Uslu, Ozlem Karsu, and B Kara. Fairness in humanitarian logistics: State of the art and future directions. *SSRN Electronic Journal*, pages 1–44, 2023.

[Fox *et al.*, 2006] Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability: Replanning versus plan repair. In Derek Long, Stephen F. Smith, Daniel Borrajo, and Lee McCluskey, editors, *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS 2006, Cumbria, UK, June 6-10, 2006*, pages 212–221. AAAI, 2006.

[Guerreiro *et al.*, 2023] Andreia P. Guerreiro, João Cortes, Daniel Vanderpooten, Cristina Bazgan, Inês Lynce, Vasco Manquinho, and José Rui Figueira. Exact and approximate determination of the pareto front using minimal correction subsets. *Comput. Oper. Res.*, 153:106153, 2023.

[Laborie *et al.*, 2009] Philippe Laborie, Jerome Rogerie, Paul Shaw, and Petr Vilím. Reasoning with conditional time-intervals. part II: an algebraical model for resources. In H. Chad Lane and Hans W. Guesgen, editors, *Proceedings of the Twenty-Second International Florida Artificial Intelligence Research Society Conference, May 19-21, 2009, Sanibel Island, Florida, USA*. AAAI Press, 2009.

[Schaus and Hartert, 2013] Pierre Schaus and Renaud Hartert. Multi-objective large neighborhood search. In Christian Schulte, editor, *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, pages 611–627. Springer, 2013.

[Schutt *et al.*, 2011] Andreas Schutt, Thibaut Feydy, Peter J. Stuckey, and Mark G. Wallace. Explaining the cumulative propagator. *Constraints An Int. J.*, 16(3):250–282, 2011.

[Senthooran *et al.*, 2023] Ilankaikone Senthooran, Matthias Klapperstück, Gleb Belov, Tobias Czauderna, Kevin Leo, Mark Wallace, Michael Wybrow, and Maria Garcia de la Banda. Human-centred feasibility restoration in practice. *Constraints An Int. J.*, 28(2):203–243, 2023.

[Smith, 2006] Barbara M. Smith. Modelling. In *Handbook of Constraint Programming*, Foundations of Artificial Intelligence, pages 377–406. Elsevier, 2006.

[Toffano *et al.*, 2022] Federico Toffano, Michele Garraffa, Yiqing Lin, Steven Prestwich, Helmut Simonis, and Nic Wilson. A multi-objective supplier selection framework based on user-preferences. *Annals of Operations Research*, pages 1–32, 2022.

[van Hoeve and Katriel, 2006] Willem-Jan van Hoeve and Irit Katriel. Global constraints. In *Handbook of Constraint Programming*, Foundations of Artificial Intelligence, pages 169–208. Elsevier, 2006.