Large neighborhood search for a multi-mode resource constrained scheduling problem with resource leveling objective

Tom Portoleau[1,2], Christian Artigues[1], Tamara Borreguero Sanchidrián[3,4], Alvaro García Sánchez[4], Miguel Ortega Mier[4] and Pierre Lopez[1]

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
{tom.portoleau,christian.artigues,pierre.lopez}@laas.fr
[2] IRIT, CNRS and University of Toulouse, France
[3] AIRBUS. Paseo John Lennon S/N. Getafe (28906) Spain
tamara.borreguero@airbus.com
[4] Industrial Engineering and Logistics Research Group, ETSII, Universidad Politécnica de Madrid. José Gutierrez Abascal 2 (28006) Madrid
{alvaro.garcia,miguel.ortega.mier}@upm.es

Keywords: multi-mode resource-constrained project scheduling, resource leveling, constraint programming, large neighborhood search.

1 Introduction

The aeronautical industry has experienced an in-depth transformation in the last years. The demand for aircrafts has increased but also the complexity and customization of the products. As a result, aircraft manufacturers have had to produce more units of more complex aircrafts while trying to reduce time to market, production lead times and costs. In line with these objectives, this paper proposes solution methods for the detailed scheduling of stations in an aircraft final assembly line. This problem, described in (Borreguero *et. al.* 2021), can be classified as a multi-mode resource-constrained project scheduling problem (MMRCPSP) with a resource leveling objective. In (Gerhards 2020) the author proposes a constraint programming (CP) model to solve the multi-mode resource investment problem (MMRIP). Formally, this problem is very similar to the assembly line scheduling problem at stake here. The notable differences are, firstly, the presence of non-renewable resources, and secondly, a slightly different objective function, as it considers a weighted sum of the maximums of the resources used, while the assembly line scheduling problem considers the maximum peak objective without weights. We propose a large neighborhood search (LNS) heuristic with neighborhoods tailored to the resource-leveling objective, with the aim to improve the performance of the CP exact methods and of the heuristic approach currently used by the company on large-scale industrial instances. As a result, the large-neighborhood search approach significantly improves the heuristic currently used by the aircraft manufacturer for assembly line scheduling. It also brings significant improvements to the method proposed by (Gerhards 2020) in the multimode resource investment problem when short CPU times are required.

2 Problem statement and constraint programming formulation

In the assembly line scheduling problem the maximum cycle time CT is fixed. We have a set of tasks W with a release date 0 and a due date CT. We have a set P of operator profiles and a set A of station areas. Each task $w \in W$ can be performed by a subset $P_w \subseteq P$ of operators profiles and by a number of operators that must lie in $M_w = \{MN_w, \ldots, MX_w\}$. The baseline duration of a task $w \in W$ is denoted by D_w. If task $w \in W$ is performed

by $o \in M_w$ operators of profile $p \in P_w$, we apply a reduction coefficient Γ_{opw} such that its duration is equal to $\Gamma_{opw} D_w$. Furthermore, each task is performed in an area indicated by $A_{aw} = 1$ if task is performed in area $a \in A$. Each area has a maximum capacity of C_a operators. There is also a set PR of standard precedence constraints such that $(w, w') \in PR$ means that w' cannot start before the end of w, a set NP of non-overlapping constraints between pairs of tasks such that $\{w, w'\} \in NP$ means that w and w' cannot be processed in parallel and a set MT of maximal time lag constraints such that $(w, w') \in MT$ means that the start time of w' must not exceed the end time of w plus a fixed time lag Δ.

We formally define the problem through the presentation of its constraint programming formulation. For modeling and solving, we use the CP Optimizer constraint-based scheduling library (Laborie *et. al.* 2018). Below we refer to the basic modeling elements we use. We refer to (Laborie *et. al.* 2018) for a more detailed definition of these elements. Each task $w \in W$ is modeled as an `interval` decision variable T_w. Each possible mode of a task (number of operators o and resource profile p) is modeled as an `interval optional` decision variable T_{wop} linked to the task by an `alternative` constraint. Precedence constraints and maximal time lag constraints are modeled by `endBeforeStart` and `startBeforeEnd` constraints, respectively. Task consumption on a resource is modeled as a `pulse` function. We use also `noOverlap` constraints for constraints related to set NP. The model comes as follows:

$\min \sum_{p \in P} n_p$
`dvar interval` T_w `in` $0..CT$, $\forall w \in W$
`dvar interval optional` T_{wop} `in` $0..CT$ `size` $\Gamma_{opw} D_w$, $\forall w \in W$, $p \in P_w$, $o \in M_w$
`alternative`$(T_w, (T_{wop})_{p \in P_w, o \in M_w})$, $\forall w \in W$
`endBeforeStart`$(T_w, T_{w'})$, $\forall (w, w') \in PR$
`startBeforeEnd`$(T_{w'}, T_w, -\Delta)$, $\forall (w, w') \in MT$
$\sum_{w \in W | A_{aw}=1} \sum_{p \in P_w} \sum_{o \in M_w}$ `pulse`$(T_{wop}, o) \leq C_a$, $\forall a \in A$
$\sum_{w \in W, o \in M_w}$ `pulse`$(T_{wop}, o) \leq n_p$, $\forall p \in P$
`noOverlap`$(T_w, T_{w'})$, $\forall \{w, w'\} \in NP$

3 A large neighborhood search approach

LNS is generally applied to scheduling problems where the objective is to minimize a time-related criterion or an outsourcing cost. A typical large neighborhood of a solution in this context consists in selecting a time interval and fixing all activities scheduled outside the interval and compacting as much as possible the activities scheduled over the interval, as it was done for the RCPSP in (Palpant *et. al.* 2004). Notably, the default search of the IBM CP Optimizer we used in the previous section also implements an LNS method based on this principle (Laborie *et. al.* 2018). This is not what we should do for the considered problem, as compacting a schedule as much as possible would inevitably increase the resource usage. In the problem considered here, we aim at minimizing the maximal use of a given resource. We aim at identifying the set of tasks that are involved in the maximal resource peaks. Consider a solution $\mathcal{S}$ where each task $w \in W$ has start time $\bar{S}_w \in [0, CT]$, a number of assigned operators $\bar{o}_w \in M_w$ for operator profile $\bar{p}_w \in P_w$ and a maximal number of operators $\bar{n}_p$ for each operator profile $p \in P$. The set of peak tasks is the set of all critical sets as defined below:

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

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

Algorithm 1 The `sweep` algorithm for peak task computation

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

In order to generate a high quality neighborhood, we let free all the tasks that contribute to the maximal use of the objective resource (the ones belonging to the peak set computed by the `sweep` algorithm) and the tasks that must precede them by a precedence constraint, and we fix the others. We then solve this new problem given the bound provided by the value of the initial solution and the constraints induced by the fixed tasks, within a limited time. If a solution has been found, it replaces the initial solution as the best solution and we start over. However, if no solution was found, we solve a new problem, fixing fewer tasks and setting a greater solving time, using a self-adaptive principle (Palpant *et. al.* 2004). To be more specific, each time the solver is unable to find a solution, we fix 10% less activities and add 10 seconds to the maximum solving time. These values were determined empirically using the instances from the benchmark considered in the previous sections. In our implementation, we use CP Optimizer as a black box to solve different generated subproblems using the constraint programming model described in Section 2. Algorithm 2 provides the pseudo-code of our implementation of the LNS method for the aircraft assembly line scheduling problem. Note that `presenceOf`(T_{wop}) in the CP Optimizer language is a constraint that enforces the presence of the optimal task T_{wop}, while `startAt`(T_w, t) is a constraint that fixes the start time of task T_w to value t. These two constraints are used to fix the modes and the start times of the tasks in $W \setminus W'$ while the tasks in W' are freed and form the LNS subproblem. Note that τ' is a value much lower than τ giving the amount of time devoted to the CP solver to get an initial solution.

4 Experimental results

We first compare LNS with CP optimizer and with a heuristic used by the aircraft manufacturer (Borreguero *et. al.* 2021) on a set of 7 industrial instances having from 90 up

Algorithm 2 LNS for the aircraft assembly line scheduling problem

Require: An aircraft assembly line scheduling problem $\mathcal{P}$ in the form of a constraint programming model (Section 2) and a time limit τ

Initialize solution $\mathcal{S}^* = \{(S^*_w, p^*_w, o^*_w)_{w\in W}, (n^*_p)_{p\in P}\}$ by solving $\mathcal{P}$ with CP Optimizer under time limit τ'

$\pi_{ratio} \leftarrow 100$; $\tau_{base} \leftarrow 10$; $\tau_{inc} \leftarrow 0$

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

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

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

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

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

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

    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \texttt{presenceOf}(T_{wo^*_w p^*_w})$

    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \texttt{startAt}(T_w, S^*_w)$

  end for

  $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{\sum_{p\in P} n_p < \sum_{p\in P} n^*_p\}$

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

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

    $\mathcal{S}^* \leftarrow \mathcal{S}$; $\tau_{inc} \leftarrow 0$; $\pi_{ratio} \leftarrow 100$

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

    $\tau_{inc} \leftarrow \tau_{inc} + 10$; $\pi_{ratio} \leftarrow \pi_{ratio} - 10$

  end if

end while

return $\mathcal{S}^*$

to 721 tasks. For a 15 min CPU time limit, LNS improves the CP optimizer solutions for 5 out of 7 instances from 5.2% to 17.6%. Compared to the heuristic used by the manufacturer, the improvement on 4 out of 7 instances goes from 4.5% to 27.7% and for one instance LNS finds a solution while the heuristic is unable to find one. We also compare LNS to the CP optimizer model proposed in (Gerhards 2020) on the MMRIP. The results displayed in Table 1 for different CPU times and numbers of tasks (from 30 to 100) show significant improvements. Detailed results are given in (Borreguero *et. al.* 2021).

Table 1. LNS improvement over CP (Gerhards 2020) for the MMRIP

Instance set	1 min	15 min	30 min
$MMRIP_{30}$	5.27%	11.15%	0.82%
$MMRIP_{50}$	5.61%	15.47%	1.14%
$MMRIP_{100}$	10.58%	17.46%	3.40%

References

Borreguero T., Portoleau T., Artigues C., García A., Ortega M., 2021, "Exact and heuristic methods for an aeronautical assembly line time-constrained scheduling problem with multiple modes and a resource leveling objective", LAAS report 21247, LAAS-CNRS, Toulouse, `https://hal.laas.fr/hal-03344445`.

Gerhards P., 2020, "The multi-mode resource investment problem: a benchmark library and a computational study of lower and upper bounds", *OR Spectrum*, Vol. 42, Num. 4, pp. 901-933.

Palpant M., Artigues C., Michelon P., 2004, "LSSPER: Solving the resource-constrained project scheduling problem with large neighbourhood search", *Annals of Operations Research*, Vol. 131, Num. 1-4, pp. 237-257.

Laborie P., Rogerie J., Shaw P.,Vilím P., 2018, "IBM ILOG CP Optimizer for scheduling", *Constraints*, Vol. 23, Num. 2, pp. 210-250.