A Polynomial-Time Algorithm for Anchored Rescheduling with Non-Availability Constraints

Pascale Bendotti[12], Luca Brunod Indrigo[12], Philippe Chrétienne[2], Bruno Escoffier[23]

[1] EDF R&D, 7 boulevard Gaspard Monge, 91120 Palaiseau, France
{pascale.bendotti, luca.brunod-indrigo}@edf.fr
[2] Sorbonne Université, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris, France
{philippe.chretienne, bruno.escoffier}@lip6.fr
[3] Institut Universitaire de France

Keywords : *Anchored rescheduling, Non-availability periods.*

1 Introduction

In most project scheduling applications, project data is subject to uncertainty. Unpredicted events may cause a previously computed *baseline schedule* to become unfeasible. In that case, a new schedule must be computed and should ideally remain close to the baseline schedule to avoid potential rescheduling costs. Furthermore, efficient algorithms are needed as rescheduling must often be done in limited time.

Anchored rescheduling consists in finding a new schedule so that a maximum number of jobs keep their starting times close to the baseline schedule. Previously introduced anchored rescheduling problems [2, 1] consider precedence constraints only. This work extends existing results by providing a polynomial solution method for a rescheduling problem with precedence constraints as well as non-availability constraints, which are essential in modelling maintenance or breakdowns for example.

2 NA-PERT

Consider a project with a set of jobs J, processing times $(p_i)_{i\in J} \in \mathbb{R}_+^J$ and precedence graph $G = (J, A)$. Every job $i \in J$ in the project is given an *NA-profile* that is a disjoint union of semi-open time intervals $U^i = \bigsqcup_{u=1}^{m_i} [a_u^i, b_u^i)$ such that $0 \leq a_1^i < b_1^i < a_2^i < b_2^i < \cdots < a_{m_i}^i < b_{m_i}^i < +\infty$. Each time interval of U^i is an *NA-period* during which job i is not available.

As stated in [5], job non-availability can be handled by considering that job completion times are functions of time. Following this idea, a starting time function $S_i : \mathbb{R}_+ \to \mathbb{R}_+$ and a completion time function $C_i : \mathbb{R}_+ \to \mathbb{R}_+$ are deduced from U^i for each job $i \in J$. The value $S_i(\tau)$ is interpreted as the earliest possible starting time of job i after or at τ. The value $C_i(\tau)$ is interpreted as the completion time of job i if it starts at $S_i(\tau)$.

The results of this work hold in particular for two classical job behaviors regarding non-availability: resumable and non-resumable. Job i satisfying the resumable execution hypothesis means that job i may be interrupted but must resume as soon as it is available again. Job i being non-resumable means that it must be processed without interruption. Starting and completion time functions are illustrated in Figure 1 for a job i with processing time $p_i = 3$ subject to two NA-periods in the resumable and non-resumable case. In Figure 1a, job i starts at $S_i(\tau) = \tau = 1$, is interrupted by the first NA-period from $a_1^i = 2$ to $b_1^i = 3$ and resumes immediately after to end at date $C_i(\tau) = 5$. In Figure 1b, there is not enough available time to process job i entirely between $\tau = 1$ and the next NA-period starting at $a_1^i = 2$, so job i starts at $S_i(\tau) = b_1^i = 3$ and ends at $C_i(\tau) = 6$.

A variant of PERT is now defined based on S_i and C_i functions.

(a) Resumable case

(b) Non-resumable case

FIG. 1: Examples of job scheduling

Definition 1 (NA-PERT) *$\mathcal{I}^U = (J, G, p, U)$ is an instance of NA-PERT if $\mathcal{I} = (J, G, p)$ is an instance of PERT and $U = (U^i)_{i \in J}$ is a vector of NA-profiles for the jobs of J. A vector $y \in \mathbb{R}_+^J$ is a schedule for $\mathcal{I}^U$ if it satisfies the following conditions:*

$$S_i(y_i) = y_i \qquad \text{for every } i \in J \tag{1}$$

$$y_j \geq C_i(y_i) \qquad \text{for every } (i,j) \in A \tag{2}$$

Condition (1) forces y_i to be a feasible starting time for job i. Condition (2) means that, if $(i, j) \in A$, the available time between the starting times of jobs i and j must be sufficient to process job i entirely. Without any NA-period, condition (2) leads to the usual precedence constraint $y_j \geq y_i + p_i$ for every $(i, j) \in A$.

3 Tolerant NA-Anchored Rescheduling problem

The rescheduling problem under study writes as follows:

Problem 1 (NA-ϵ-Anchored Rescheduling) ***Input:*** *An instance $\mathcal{I}^U = (J, G, p, U)$ of NA-PERT, a baseline schedule $x \in \mathbb{R}_+^J$, a tolerance vector $(\epsilon_i)_{i \in J} \in \mathbb{R}_+^J$.*
Question: *Find a schedule y of $\mathcal{I}^U$ that maximizes $|\{i \in J \mid |x_i - y_i| \leq \epsilon_i\}|$.*

The core result is that NA-ϵ-Anchored Rescheduling can be efficiently solved.

Theorem 1 *NA-ϵ-Anchored Rescheduling can be solved in polynomial time.*

Theorem 1 is proved by reducing NA-ϵ-Anchored Rescheduling to finding a maximum antichain in an appropriate poset, which is a polynomial problem [4]. Similarly, an extension of Theorem 1 to a weighted version of NA-ϵ-Anchored Rescheduling is possible by reducing it to a maximum weight antichain problem, which is also a polynomial problem [3].

References

[1] Pascale Bendotti, Philippe Chrétienne, Pierre Fouilhoux, and Adèle Pass-Lanneau. Anchored rescheduling problems under generalized precedence constraints. In *International Symposium on Combinatorial Optimization. Lecture Notes in Computer Science*, volume 12176, pages 156–166. Springer, 2020.

[2] Pascale Bendotti, Philippe Chrétienne, Pierre Fouilhoux, and Alain Quilliot. Anchored reactive and proactive solutions to the cpm-scheduling problem. *European Journal of Operational Research*, 261(1):67–74, 2017.

[3] Kathie Cameron. Antichain sequences. *Order*, 2(3):249–255, 1985.

[4] Robert P Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–166, 1950.

[5] Michel Minoux. Models and algorithms for robust PERT scheduling with time-dependent task durations. *Vietnam Journal of Mathematics*, 35, 2007.