

The Continuous Time-Resource Trade-off Scheduling Problem with Time Windows

Christian Artigues, Emmanuel Hébrard, Alain Quilliot, Hélène Toussaint

▶ **To cite this version:**

Christian Artigues, Emmanuel Hébrard, Alain Quilliot, Hélène Toussaint. The Continuous Time-Resource Trade-off Scheduling Problem with Time Windows. INFORMS Journal on Computing, 2024, 36 (6), pp.1359-1756. ⟨10.1287/ijoc.2022.0142⟩. ⟨hal-04610399⟩

HAL Id: hal-04610399

https://hal.science/hal-04610399v1

Submitted on 10 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

The continuous time-resource tradeoff scheduling problem with time windows

Christian Artigues[1], Emmanuel Hébrard[1], Alain Quilliot[2], and Hélène Toussaint[2]

[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[2]LIMOS CNRS/UCA, Clermont-Ferrand France

October 10, 2024

Abstract

We introduce a variant of the cumulative scheduling problem (CuSP), characterized by continuous modes, time windows and a criterion which involves safety margin maximization. The study of this variant is motivated by the GEOSAFE H2020 project, devoted to the design of evacuation plans in face of natural disasters and more specifically wildfire. People and goods have to be transferred from endangered places to safe places and evacuation planning consists in scheduling evacuee moves along pre-computed paths, under arc capacities and deadlines. The resulting model is relevant in other contexts, such as project or industrial process scheduling. We consider here several formulations of the continuous time-resource tradeoff scheduling problem (CTRTP-TW) with a safety maximization objective. We establish a complete complexity characterization distinguishing polynomial and NP-hard special cases depending on key parameters. We show that the problem with fixed sequencing, i.e. with predetermined overlap or precedence relations between activities, is convex. We then show that the preemptive variant is polynomial and we propose lower and upper bounds based on this relaxation. A flow-based mixed-integer linear programming formulation is presented, from which a branch-and-cut exact method and an insertion heuristic are derived. An exact dedicated branch-and-bound algorithm is also designed. Extensive computational experiments are carried out to compare the different approaches on evacuation planning instances and on general CTRTP-TW instances. The experiments show also the interest of the continuous model compared to a previously proposed discrete approximation.

1 Introduction

The NP-hard cumulative scheduling problem (CuSP) [11] can be stated as the problem to find a feasible schedule for a set of activities that share a single so-called cumulative resource with a finite capacity. Each activity is associated with a time-window and a fixed amount of resource required during all its processing interval. A schedule is feasible if each activity is scheduled within its time window and if, at each time point, the total amount of resource required by the activities in progress never exceed the resource capacity.

The CuSP is at the core of the well known resource constrained project scheduling Problem (RCPSP, see [1, 14, 21, 23, 26, 32] for surveys), one of the scheduling problems that received the most attention during the last decades. The RCPSP involves a set of activities subject to precedence and resource constraints: each activity requires simultaneously during its processing

interval a different amount on each of the cumulative resources. A performance criterion has to be optimized, which is most often the makespan (total duration) minimization. Many variants of the RCPSP exist, some of which being related to preemption (some activities may be run, interrupted, and run again), non renewable resources or alternative performance criteria.

Among those variants, the multimode RCPSP [9, 10, 32, 36, 39, 41]: consider activities with flexible resource requirements and the decision maker may cut a well-fitted trade-off between resource consumption and the durations of the activities. The possible modes are most often discrete: an activity mode gives a precise duration, a list of required resources, and, for each of them, a required amount.

The CTRTP-TW (continuous time-resource tradeoff problem with time windows) model introduced in this paper can be viewed as a variant of CuSP that involves multiple resources and modes: for any activity j, we have to choose its processing rate s_j in an interval $[s_j^{\min}, s_j^{\max}]$, which determines, for any resource k in the set $\mathcal{R}_j$ of the resources required by j, the amount of k consumed by j. For a given processing rate s_j, the duration of the activity is $p_j = P_j/s_j$, where P_j is the work content of activity j. As for temporal constraints, we have to select for each activity j, a start time S_j and a completion time C_j subject to release dates r_j and deadlines d_j. The objective is safety maximization and more precisely the maximization of the minimum safety margin of every activity, where the safety margin of an activity denotes the difference between its deadline and its ending time. This maximization of this objective is equivalent to minimizing the maximum lateness. Compared to the standard multimode RCPSP with continuous resources (see chapter 10 in [32] for a precise definition), the CTRTP-TW has the following specificities: the activities have no precedence constraints but release dates and deadlines; once an activity is started, its processing rate remains constant till the end of the activity; the processing rate of each activity must lie within a predefined interval.

Among the vast literature on special cases and variants of the multimode RCPSP, the considered problem is close to the discrete time-resource tradeoff problem (DTRTP), see for instance [13]. However, in the CTRTP-TW, the function that links the duration of an activity to its procesing rate s_j is the rational function P_j/s_j instead of a discrete function defined on a discrete set as in the DTRTP. The feasibility variant of the CTRTP with a single resource was defined as the cumulative resource constraint with energy in [42], with, additionally, a variable total work requirement. Constraint propagation algorithms were proposed in a discrete time setting. Considering discrete time and/or resource requirements may not accurately model particular contexts and yield infeasible or suboptimal solutions.

The CTRTP (where both resource and time are continuously divisible) has been previously considered in the literature. In [33], the major difference is that the duration function is approximated as an affine linear function $p_j = b_j - a_j s_j$, where b_j is the maximal duration of the activity and a_j is obtained by linear regression. So the problem can be addressed via a mixed-integer linear programming approach inspired by rectangle packing. In our case, the nonlinear and continuous model is such that rational solutions may not exist as we will show on a simple example.

The CTRTP with general processing rate functions is considered in several papers such as [40, 35]. The processing rate function is the function $f(.)$ that links the total work content to the processing rate, i.e. $P_j = \int_{t=S_j}^{C_j} f_j(r_j)dt = (C_j - S_j)f_j(r_j)$. In the context considered in papers [35, 40], the case where f is convex, is trivially solved. In our case we have also a convex function, more precisely the identity function $f(x) = x$, but due to the presence of release dates and deadlines, the problem will be shown to be NP-hard.

In [27, 28, 38], a problem called the resource-constrained project scheduling problem with flexible resource profiles (FRCPSP) involving the assignment of continuous resources is considered. In the FRCPSP, the processing rate r_i is not constant during the processing window of

an activity and may differ on each required resource $k \in \mathcal{R}_j$. This means that a functional $t \to r_{j,k}(t)$ has to be determined for each activity $j \in \mathcal{J}$ and each resource $k \in \mathcal{R}_j$ such that $P_j \leq \int_{t=S_j}^{C_j} f_j(r_j(t))dt$. In [28, 38], the authors propose time-indexed MILP models defined on a discrete time horizon. In [27], event-based MILP formulation is devised in a continuous time setting. In the FRCPSP, there are precedence constraints but the problem is to minimize the makespan and the activities have no time windows. Each activity $j \in \mathcal{J}$ has a principal resource $k^* \in \mathcal{R}_j$ such that the processing rate $r_{j,k}(t)$ of an activity $j \in \mathcal{J}$ on a resource $k \in \mathcal{R}_j \setminus k^*$ at each time period t depends linearly on $r_{j,k^*}(t)$. A model close to the FRCPSP, the continuous energy-constrainted scheduling problem (CECSP), is considered in [6, 29, 30], with a single resource and activity time windows. The major difference between the CTRTP-TW and the FRCPSP/CECSP models is that in the CTRTP-TW, the processing rate is a variable that must keep the same value from the start to the end of an activity. In this paper we exploit this strong characteristic with dedicated algorithms.

The study of the CTRTP-TW was motivated by a work carried out in the context of the H2020 GEO-SAFE European project [19]. The overall objective of GEO-SAFE was to develop methods and tools enabling to set up an integrated decision support system to assist authorities in optimizing the resources during the response phase to some wildfire. In such a circumstance, decisions to be taken aim at fighting the cause of the disaster, adapting standard logistics (food, drinkable water, health,...) to the current state of infrastructures and evacuating endangered areas [34]. The problem related to the CTRTP-TW is the Late Evacuation problem (LEP [2]). The LEP consists in computing an evacuation schedule for people (and possibly critical goods), which had been staying at their location as long as possible (late evacuees). While in practice evacuation planning is mainly managed in an empirical way, optimization approaches have been proposed [2, 3, 8, 16, 31, 22] according to a 2-step process: the first step (pre-process) involves the identification of the routes that evacuees are going to follow; the second step, which has to be performed in real time, aims at scheduling the evacuation of estimated late evacuees along those routes. This last step involves 2 distinct work pieces, one about forecasting, which is difficult in the case of wildfires because of their dependence on topography and meteorology [34], and the second one about the specification of priority rules and evacuation rates imposed to evacuees [8, 15]. We only deal with this last issue and do not address forecasting. In order to ensure the robustness of the evacuation process, practitioners require evacuees to be clustered into groups with same original locations and pre-computed routes, and they do in such a way that these routes define a tree. Besides, they impose that once a group starts moving, it keeps on at the same evacuation rate until reaching his target safe area. It can be remarked that resulting LEP models may be cast into the CTRTP-TW framework. Such an approach was initiated in [16] in the context of evacuations due to floodings, with the difference that the activity rates and durations where constrained to be integer, which yielded the DTRTP model. In the context of the GEOSAFE project, the same model (with integer variables) was used to represent evacuation problems in case of wildfire [2, 3]. Under the same context, the CTRTP-TW framework was introduced for the first time in [31]. Using continuous (rational) variables instead of integer ones has modeling benefits. The paper proposed a flow-based formulation for the problem.

Furthermore, one can also remark that CTRTP-TW may also arise in industrial contexts, when assembly processes involving pipelining mechanisms have to be scheduled [12, 21, 23].

In this paper we prove that even special cases of the CTRTP-TW are NP-hard while other relaxations, including the preemptive one, can be solved in polynomial time. We deal with CTRTP-TW through both a heuristic algorithm implementings fast network flow techniques, well-fitted to real-time management, and an exact branch-and-bound algorithm relying on upper bounds derived from the exact characterization of optimal preemptive solutions.

The paper is structured as follows: Section 2 describes the general CTRTP-TW model and provides a complexity analysis. Section 3 discusses a first formulation of CTRTP-TW relying on a bi-level decomposition of the problem into a combinatorial master level, which addresses the sequencing of the activities and a convex slave level, which deals with continuous evacuation rates. Section 4 addresses the preemptive CTRTP-TW, shows that it can be solved in polynomial time through a sequence of rational linear programs, and derives upper bounds as well as relaxation-induced heuristics for the non-preemptive CTRTP-TW. Section 5 proposes a second, network flow-oriented, reformulation of the CTRTP-TW and describes a branch-and-cut exact method and a flow-based heuristic. A dedicated branch-and-bound algorithm proposed in Section 6. Section 7 is devoted to numerical tests involving late evacuation planning instances as well as general CTRTP-TW instances. Note that the paper is an extended version of the short paper [4].

2 The CTRTP-TW: continuous time-resource tradeoff scheduling problem with time windows and safety maximization

We give a formal model of CTRTP-TW in Section 2.1. Complexity results are subsequently established (Section 2.2).

2.1 A formal CTRTP-TW model

The CTRTP-TW takes as input a set $\mathcal{J}$ of activities and a set $\mathcal{R}$ of resources. Each activity $j \in \mathcal{J}$ is characterized by a release date r_j, a deadline d_j, a total work content P_j, a minimum processing rate $s_j^{\min}$ and a maximum processing rate $s_j^{\max}$. Each resource $k \in \mathcal{R}$ has a capacity R_k and a set of activities $\mathcal{J}_k \subseteq \mathcal{J}$ that must be processed by the resource. We define the subset of resources required by activity j as $\mathcal{R}_j = \{k \in \mathcal{R} | j \in \mathcal{J}_k\}$.

The output of the problem is fully specified by a start time S_j and a processing rate (speed) s_j, for each activity $j \in \mathcal{J}$, that maximize objective (1) and satisfy constraints (2–6).

$$\text{maximize} \min_{j \in \mathcal{J}} (d_j - C_j) \tag{1}$$

$$\text{s.t.} \quad C_j = S_j + P_j / s_j \qquad \forall j \in \mathcal{J} \tag{2}$$

$$S_j \geq r_j \qquad \forall j \in \mathcal{J} \tag{3}$$

$$C_j \leq d_j \qquad \forall j \in \mathcal{J} \tag{4}$$

$$s_j^{\min} \leq s_j \leq s_j^{\max} \qquad \forall j \in \mathcal{J} \tag{5}$$

$$\sum_{\substack{j' \in \mathcal{J}_k \\ S_{j'} < C_j, S_j < C_{j'}}} s_{j'} \leq R_k \qquad \forall k \in \mathcal{R}, \forall j \in \mathcal{J}_k \tag{6}$$

To simplify the expression, the formulation uses the redundant completion time C_j variable. Objective (1) consists in maximizing the safety margin. Constraints (2) express that the duration of each activity is equal to the total work content divided by the processing rate (speed). Constraints (3) and (4) force each activity to be scheduled within its time window. Constraints (5) set lower and upper bounds for the processing rate of each activity. Resource constraints (6), state that when an activity starts, the cumulated processing rate of the activities that share the same resources and that are in progress cannot exceed the resource capacity.

It must be remarked that to be able to satisfy the total work content constraint (2) we must have

$$s_j \geq P_j / (d_j - r_j) \quad \forall j \in \mathcal{J}$$

This allows to adjust $s_j^{\min}$ accordingly.

Example 1. *To give some insight on the problem structure, consider a simple example with 2 activities, a single resource of capacity 2, and the following parameters:* $r_1 = 3$, $P_1 = 6$, $d_1 = 11$, $r_2 = 4$, $P_2 = 4$, $d_2 = 8$, $s_1^{\min} = s_2^{\min} = 0$, $s_1^{\max} = 1$, *and* $s_2^{\max} = 2$. *We can solve analytically the problem by first remarking that there are only three possibilities for the relative sequencing of the two activities. If activity* 1 *is scheduled before activity* 2 *then, at the earliest, activity* 2 *starts at time* $r_1 + P_1/s_1^{\max} = 9$ *and ends at* $9 + P_2/s_2^{\max} = 12$, *which violates the deadline of activity* 2. *Conversely if activity* 1 *is scheduled after activity* 2, *then activity* 1 *cannot start before* $r_2 + P_2/s_2^{\max} = 6$ *and end before* $6 + P_1/s_1^{\max} = 12$, *which also violates the deadline of activity* 1. *Consequently, to obtain a positive safety margin. activities* 1 *and* 2 *must overlap. In this condition, there also exists an optimal solution such that each activity is scheduled at its release date. Otherwise, one could left-shift the activity that does not start at its release date without increasing the minimum margin until the release date is reached. Similarly, we can show that there exists an optimal solution such that rates* s_1 *and* s_2 *saturate the resource capacity (i.e.* $s_1 + s_2 = R_k$*). Otherwise, one could increase the rate of activity* 2, *which is bounded by the capacity until the capacity is saturated, which would reduce the duration of activity* 2, *hence increasing its margin. Given these assumptions we can now write the problem as follows:*

$$\begin{aligned} \textit{maximize} \quad & \lambda = \min(d_1 - r_1 - P_1/s_1, d_2 - r_2 - P_2/s_2) \\ \textit{s.t.} \quad & s_1 \leq 1 \\ & s_1 + s_2 = 2 \end{aligned}$$

Hence by replacing s_2 *by* $2 - s_1$, *we obtain the objective function* $\min(8 - 6/s_1, 4 - 4/(2 - s_1))$. *For* $s_1 \in (0, 1]$, *the left term is an increasing function from* $-\infty$ *to* 2 *while the right term is a decreasing function from* 2 *to* 0. *It follows that the maximum of the minimum of the two terms is attained at the equality, which yields the following polynomial equation:*

$$2s_1^2 - 9s_1 + 6 = 0$$

The only root of this polynomial in $(0, 1[$ *is* $s_1^* = \frac{9-\sqrt{33}}{4} \simeq 0.81$, *which yields* $s_2^* = \frac{\sqrt{33}-1}{4} \simeq 1.19$ *and* $\lambda^* = \frac{48-8\sqrt{33}}{9-\sqrt{33}} \simeq 0.63$.

Example 2. *Consider a second example with* $\mathcal{J} = \{1, 2, 3\}$, $P = \{3, 3, 4\}$, $r = \{3, 3, 4\}$, $d = \{13, 13, 7\}$, *and 2 resources of capacities* $R = \{1, 2\}$ *such that* $\mathcal{J}_1 = \{1, 2\}$ *and* $\mathcal{J}_1 = \{1, 2, 3\}$. *We get an optimal schedule by starting activity 3 at time* $S_3 = 4$ *according to full rate* $s_3 = 2$, *yielding* $C_3 = 6$. *Activities 1 and 2 both start at* $S_1 = S_2 = 6$ *with rate* $s_1 = s_2 = 1/2$ *and complete at time* $C_1 = C_2 = 12$. *We observe a minimum satefy margin of* $d_1 - C_1 = d_2 - C_2 = d_3 - C_3 = 1$. *This optimal schedule is displayed in Figure 1.*

Example 3. *Let us now change example 2 by considering different deadlines* $d = \{9, 9, 8\}$. *An optimal schedule is obtained by reducing the rate of activity 3 to 1, which allows to start the evacuation of activities 1 and 2 at time 3 at the same rate (see Figure 2). Now the minimum safety margin reduces to* $d_1 - C_1 = d_2 - C_2 = d_3 - C_3 = 0$.

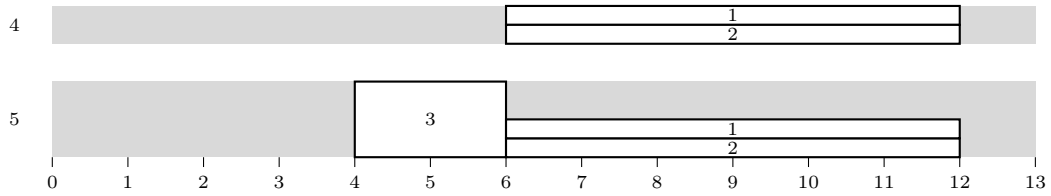


Figure 1: Optimal solution of CTRTP-TW example 2

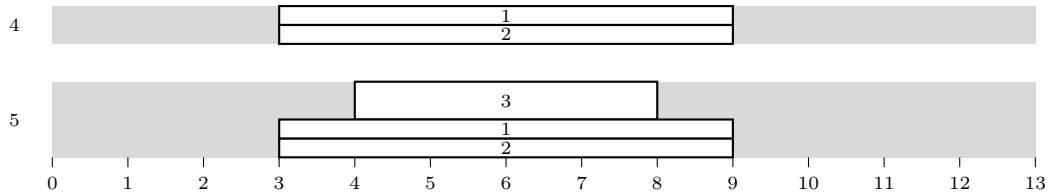


Figure 2: Optimal solution of CTRTP-TW example 3

2.2 Complexity of CTRTP-TW with a single resource

Here we consider the case where there is a unique resource k shared by all activities and, in particular, its decision version of whether there exists a solution of cost less than some value λ. We show exactly which constraints on start times and evacuation rates make this particular case NP-complete or polynomial. We denote S-CTRTP-TW(C) the single resource CTRTP-TW with constraints C where $C \subseteq \{r, d, s^{\min}, s^{\max}\}$ with the following meaning :

- $r \notin C$ means that the release dates are all equal to 0.
- $d \notin C$ means that the problem aims at minimizing the makespan
- $s^{\min} \notin C$ means that the minimum rates are all equal to 0.
- $s^{\max} \notin C$ means that the maximum rates are all equal to R_k.

Theorem 1. *S-CTRTP-TW(C) is strongly NP-hard if* $|C| \geq 2$ *and* $C \neq \{x, s^{\min}\}$ *with* $x \in \{r, d\}$

Proof. We first show that the decision S-CTRTP-TW, which asks if there exists a solution of at least λ, is in NP. Indeed, once the start time S_j, the completion time C_j, and the rate s_j have been fixed for each activity, the time window constraints (3) and (4), the rate limitation constraints (5), the duration constraints (2), and the objective function value (1) can be checked in $O(|\mathcal{J}|)$ time. The resource constraints (6) can be checked in $O(|\mathcal{J}|^2|\mathcal{R}|)$ time by enumerating, for each resource, the start times of every activity and then by computing the cumulated rate of activities on the same resource, which overlap with this time point.

The fact that S-CTRTP-TW(C) is $NP-$hard if $\{s^{\min}, s^{\max}\} \subseteq C$ can be easily shown by remarking that any instance of the parallel machine problem $P||C_{\max}$ can be reduced to a S-CTRTP-TW($\{s^{\min}, s^{\max}\}$) instance as follows. We have n activities of duration p_i to be scheduled on m identical machines. By simply setting $R_k = m$ and, for each activity $j = 1, \ldots, n$, $s_j^{\min} = s_j^{\max} = 1$ and $P_j = p_i$, we obtain the parallel machine problem, which is strongly NP-hard. We could also use the CuSP to obtain the same result.

Now we show that S-CTRTP-TW(C) is $NP-$hard when $\{x, y\} \subseteq C$ with $s^{\min} \notin \{x, y\}$ by reduction from the 3-Partition problem. Let M be an integer, $n = (n_1, \ldots, n_N)$ with $N = 3M$

be a non negative integral vector, and $M \cdot T = \sum_{q=1}^{N} n_q$. The 3-PARTITION problem on instance n is to decide whether there is a partition of n into M triplets of equal sum. This problem is known to be strongly NP-complete even if every element is strictly between $T/4$ and $T/2$ [18]. We consider this restriction of the problem and we show that it can be polynomially reduced to any S-CTRTP-TW(x, y) instance with $s^{\min} \notin \{x, y\}$.

For that purpose, we build an S-CTRTP-TW instance as follows. We define a single resource ($\mathcal{R} = \{k\}$). The capacity of k is $R_k = 3T$ and an activity set $\mathcal{J} = \{1, \ldots, N\} \cup \{sep_1, \ldots, sep_{M-1}\} \cup \{fill_1, \ldots, fill_M\}$ with:

- For all $j = 1, \ldots, N$, $P_j = n_j$.
- For all $i = 1, \ldots, M$, $P_{sep_i} = 3T$
- For all $i = 1, \ldots, M-1$, $P_{fill_i} = 2T$

Furthermore, depending on the problem type we define the additional constraints:

- If $(x, y) = (r, d)$, we set $r_j = 0$, $d_j = M$, $\forall j = 1, \ldots, N$, $r_{sep_i} = 2i - 1$, $d_{sep_i} = 2i$, $\forall i = 1, \ldots, M-1$, $r_{fill_i} = 2i - 2$, $d_{fill_i} = 2i - 1$, $\forall i = 1, \ldots, M-1$.
- If $(x, y) = (r, s^{\max})$, we set $r_j = 0$, $s_j^{\max} = T$, $\forall j = 1, \ldots, N$, $r_{sep_i} = 2i - 1$, $s_{sep_i}^{\max} = 2T$, $\forall i = 1, \ldots, M-1$, $r_{fill_i} = 2i - 2$, $s_{fill_i}^{\max} = 3T$, $\forall i = 1, \ldots, M-1$.
- If $(x, y) = (s^{\max}, d)$, we set $d_j = M$, $s_j^{\max} = T$, $\forall j = 1, \ldots, N$, $d_{sep_i} = 2i$, $s_{sep_i}^{\max} = 2T$, $\forall i = 1, \ldots, M-1$, $d_{fill_i} = 2i - 1$, $s_{fill_i}^{\max} = 3T$, $\forall i = 1, \ldots, M-1$.

We want to schedule these activities while achieving a zero safety margin within a makespan of $2M-1$. Suppose first that we have a solution of the 3-PARTITION problem. It follows that we can partition the set of activities $\mathcal{J}$ into M triplets $\mathcal{J}_1, \ldots, \mathcal{J}_M$ such that $\forall i \in \{1, \ldots, M\}$, $\sum_{j \in J_i} P_j = T$. We show that we can then obtain a feasible schedule as follows:

- For each $i \in \{1, \ldots, M-1\}$, we set $S_{sep_i} = 2i - 1, C_{sep_i} = 2i$, and $s_{sep_i} = 3T$;
- For each $i \in \{1, \ldots, M\}$, we set $S_{fill_i} = 2i - 2, C_{fill_i} = 2i - 1$, and $s_{fill_i} = 2T$;
- For each $i \in \{1, \ldots, M\}$, let $J_i = \{a, b, c\}$, we set $s_a = s_b = s_c = T$, $S_a = 2i - 2$, $C_a = S_b = 2i - 2 + a/T$, $C_b = S_c = 2i - 1 - c/T$, and $C_b = 2i - 1$.

This solution is illustrated in Figure 3 and one can check that none of the additional constraints for any considered $\{x, y\}$ case is violated by this solution.

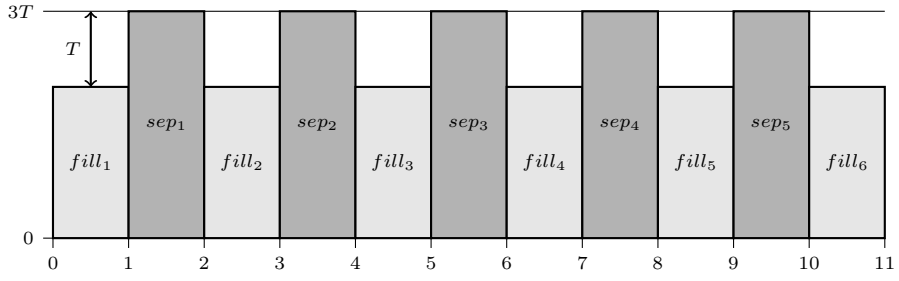


Figure 3: Illustration of the reduction of SubsetSum to CTRTP-TW

Conversely, we show that given a feasible schedule, we can build a solution of the 3-PARTITION problem. There are three remaining cases for the pair (x, y), we first show that the activities $sep_1, \ldots, sep_{M-1}$ and $fill_1, \ldots, fill_M$ are scheduled in the same way in all cases:

- If $(x, y) = (r, d)$, because of the release dates and deadlines, the activities $sep_1, \ldots, sep_{M-1}$ and $fill_1, \ldots, fill_M$ have no freedom, their start times must be equal to their release dates and their completion times to their deadlines. It follows that their rates are all equal to $3T$ and $2T$, respectively, and hence no other task can be processed on intervals of the form $[2i-1, 2i]$.
- If $(x, y) = (r, s^{\max})$, we use an induction with the following hypothesis: $S_{fill_i} = 2i-2$, $s_{fill_i} = 2T$, $S_{sep_{i-1}} = 2i-3$, and $s_{sep_i} = 3T$. Activity $fill_M$ can only end before its deadline if its rate is maximum and if it starts at its release date. Now for sep_{M-1} to overlap with $fill_M$, it must have a rate less than or equal to T and hence a start time less than or equal to $2M-4$, which is a contradiction. Since it does not overlap with sep_{M-1}, it must have a rate of $3T$ and start at $2M-3$. Now suppose that the induction is true for all $i > k$. By the induction hypothesis, on the interval $[2k-1, 2M-1]$, no activity of work content greater than T can be executed, hence activity $fill_k$ must start before $2k-1$. Moreover, since activity sep_k has rate $3T$, any activity that starts before $2k-1$ must be finished at $2k-1$. Since $r_{fill_k} = 2k-2, P_{fill_k} = 2T$, and $s^{\max}_{fill_k} = 2T$, we have $S_{fill_k} = 2k-2$ and $s_{fill_k} = 2T$. With a similar argument, we have $S_{sep_{k-1}} = 2k-3$ and $s_{sep_{k-1}} = 2T$.
- if $(x, y) = (s^{\max}, d)$, the reasoning is symmetric to the previous case.

Notice that the instance is built in such a way that a solution does not waste any time or resource. Therefore, in all cases and for all $i \in \{1, \ldots, M\}$, the activities processed on the interval $[2i-2, 2i-1]$ have a total work content of $3T$. Since job $P_{fill_i} = 2T$, the jobs $\{1, \ldots, N\}$ processed on this interval have a total work content of T. Moreover, there must be exactly three of them since $T/4 < P_j < T/2, \forall i \in \{1, \ldots, N\}$, and hence a feasible schedule corresponds to a 3-partition. □

We now show that all the other cases are polynomial.

Theorem 2. *S-CTRTP-TW($s^{\max}$), S-CTRTP-TW($r, s^{\min}$), S-CTRTP-TW($d, s^{\min}$) can all be solved in polynomial time*

Proof. For S-CTRTP-TW($s^{\max}$), the minimum duration of each activity is $P_j / s^{\max}_j$. The problem is infeasible if $P_j / s^{\max}_j > \lambda$ for some activity j. Otherwise we set the rate of each activity to λ / P_j and each start time $S_j = 0$. The problem is feasible if and only if $\sum_{j \in \mathcal{J}} \lambda / P_j \leq R_k$.

For S-CTRTP-TW($r, s^{\min}$), S-CTRTP-TW($d, s^{\min}$), release dates and deadlines clearly play a symmetric role. So we give here the proof for S-CTRTP-TW($s^{\min}, d$). Since there is no maximum rate, the rate of each activity can be set to $s_j = R_k$ and we obtain the decision variant of the one-machine problem (denoted $1||L_{max}$ in the standard three-field scheduling notation) that can be solved in $O(|\mathcal{J}| \log(|\mathcal{J}|))$ with the EDD rule. □

Table 1 synthesizes the complexity results for the S-CTRTP-TW, leaving no open cases.

3 CTRTP-TW with fixed sequencing

3.1 A bilevel formulation of CTRTP-TW

We first give a bilevel formulation for CTRTP-TW equivalent to formulation (1–6) introducing a binary sequencing variable $\pi_{j_1, j_2} \in \{0, 1\}$, for each pair of activities $j_1, j_2 \in \mathcal{J}$ such that $\pi_{j_1, j_2} = 1$ if and only if precedence relation $C_{j_1} \leq S_{j_2}$ holds. It follows that two activities overlap if and

Constraints	Complexity
$s^{\max}$	P
$r, s^{\min}$	P
$d, s^{\min}$	P
$s^{\min}, s^{\max}$	strongly NP-complete
$r, s^{\max}$	strongly NP-complete
$d, s^{\max}$	strongly NP-complete
r, d	strongly NP-complete

Table 1: Complexity map of S-CTRTP-TW

only if $\pi_{j_1,j_2} = \pi_{j_2,j_1} = 0$. Now, given a fixed sequencing π, a slave problem CTRTP-TW(π) can be written as :

$$\text{maximize} \quad \min_{j \in \mathcal{J}}(d_j - C_j) \tag{7}$$

$$C_j \geq S_j + P_j/s_j \qquad \forall j \in \mathcal{J} \tag{8}$$

$$S_{j_2} \geq C_{j_1} \qquad \forall j_1, j_2 \in \mathcal{J}, \pi_{j_1,j_2} = 1 \tag{9}$$

$$S_j \geq r_j \qquad \forall j \in \mathcal{J} \tag{10}$$

$$C_j \leq d_j \qquad \forall j \in \mathcal{J} \tag{11}$$

$$s_j^{\min} \leq s_j \leq s_j^{\max} \qquad \forall j \in \mathcal{J} \tag{12}$$

$$\sum_{j \in \mathcal{I} \cap \mathcal{J}_k} s_j \leq R_k \qquad \forall k \in \mathcal{R}, \forall \mathcal{I} \subseteq \mathcal{J} \text{ such that } \forall j_1, j_2 \in \mathcal{I}, \pi_{j_1,j_2} = \pi_{j_2,j_1} = 0 \tag{13}$$

Let $z(\pi)$ denote the optimal solution of CTRTP-TW(π). Let Π denote the set of valid sequencings. Then CTRTP-TW can be defined as the master problem to find the sequencing π^* that maximizes the maximin safety margin with fixed sequencing as follows:

$$z(\pi^*) = \max_{\pi \in \Pi} z(\pi)$$

3.2 A compact convex program

Theorem 3. *CTRTP-TW with fixed sequencing is convex.*

Proof. It comes in a straightforward way from the fact that function $x \to \frac{1}{x}$ is convex □

Despite the fact that CTRTP-TW(π) is convex, it remains that there is in general an exponential number of constraints (13) due to the enumeration of all subsets of overlapping activities. Let us consider the undirected graph $\tilde{\mathcal{G}}(\pi)$ with a node for each activity and an edge linking two nodes such that the corresponding activities j_1 and j_2 satisfy the overlapping relation $\pi_{j_1,j_2} = \pi_{j_2,j_1} = 0$. It is trivially verifiable that the set of subsets $\mathcal{I} \subseteq \mathcal{J}$ such that $\forall j_1, j_2 \in \mathcal{I}, \pi_{j_1,j_2} = \pi_{j_2,j_1} = 0$ maps the set of cliques in this graph. It is also easy to notice that only the constraints linked to the maximal cliques of the $\tilde{\mathcal{G}}(\pi)$ need be considered. In general, in an n-node graph, there is an exponential number of cliques that can be optimally enumerated in $O(3^{n/3})$ time by the Bron–Kerbosch algorithm [37]. However, we remark that for the sequencing π derived from any feasible solution to the CTRTP-TW, $\tilde{\mathcal{G}}(\pi)$ is an interval graph. The set of valid sequencings Π can even be mapped to the set of $|\mathcal{J}|$-node interval graphs, which has a number of maximal cliques not greater than $|\mathcal{J}| + 1$ that can be enumerated in $O(|\mathcal{J}| \log(|\mathcal{J}|))$ [20]. Hence under the restriction that $\tilde{\mathcal{G}}(\pi)$ is an interval graph, which can be checked in $O(|\tilde{\mathcal{G}}(\pi)|)$, $|\tilde{\mathcal{G}}(\pi)|$ denoting

the sum of edge and vertices numbers, CTRTP-TW(π) is a compact convex program, i.e. with a polynomial number of variables and constraints.

4 Upper and lower bounds based on the preemptive case

We now consider the preemptive relaxation P-CTRTP-TW in which any activity $j \in \mathcal{J}$ can be interrupted at any time and restarted later. Its rate has to remain lower than $s_j^{\max}$ but the minimum rate constraint is also relaxed. Solving P-CTRTP-TW yields obviously an upper bound for CTRTP-TW.

Example 4. *Let us change again the deadlines in Example 2 instance by $d_1 = d_1 = 12$ and $d_3 = 7$. The preemptive solution displayed in Figure 4 has a safety margin of 1 unit, which is not reachable with a non-preemptive solution. Indeed the optimal non-preemptive solution, displayed in Figure 5, is reached with $S_1 = S_2 = 3$, $S_3 = 4$, $s_1 = s_2 = 1/3$, and $s_3 = 4/3$ for a zero safety margin.*

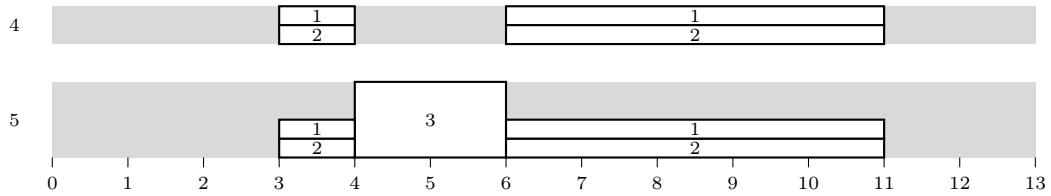


Figure 4: P-CTRTP-TW optimal solution for $r_1 = r_2 = 3$, $r_3 = 4$, $d_1 = d_2 = 12$, $d_3 = 7$

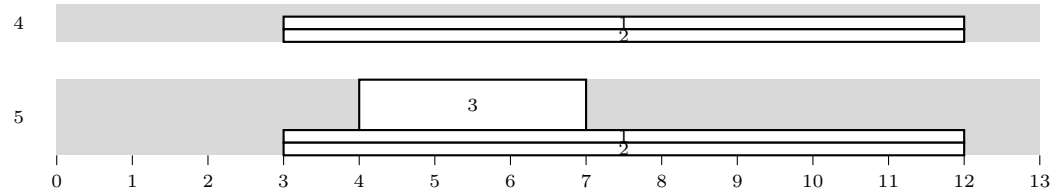


Figure 5: (Non-preemptive) CTRTP-TW optimal solution for $r_1 = r_2 = 3$, $r_3 = 4$, $d_1 = d_2 = 12$, $d_3 = 7$

4.1 Solving P-CTRTP-TW

We consider first the decisional problem P-CTRTP-TW(λ) of finding a preemptive schedule having a safety margin of at least λ. We show that P-CTRTP-TW can be solved in polynomial time by linear programming using an extension of the max-flow formulation proposed in [24] for the decision variant of the preemptive parallel machine problem with release dates and maximum lateness minimization (denoted $P|r_i|L_{\max}$ in the standard three-field scheduling notation). We introduce for each activity $j \in \mathcal{J}$ a deadline $\tilde{d}_j = d_j - \lambda$. Then we consider the ordered set $\mathcal{T} = \{t_1, \dots t_{|\mathcal{T}|}\}$ of distinct release dates and deadlines.

We remark that any preemptive schedule can be changed into a schedule where every activity has a constant rate for $t_{q+1} - t_q$ time units in every interval $[t_q, t_{q+1}]$ with $q \in \{1, \dots |\mathcal{T}| - 1\}$

in which it is processed. This is due to the relaxation of the minimum rate constraints and to the absence of precedence constraints between the activities. Consider now the following linear program:

We define a continuous variable $s_j^q \geq 0$ that represents the constant rate of activity j during interval $[t_q, t_{q+1}]$, $q \in Q$. Then P-CTRTP-TW(λ) amounts to finding a solution to the following a system of linear equations and inequalities, which can be solved with an LP solver.

$$\sum_{q \in Q} (t_{q+1} - t_q) s_j^q = P_j \qquad \forall j \in \mathcal{J} \tag{14}$$

$$\sum_{j \in \mathcal{J}_k} s_j^q \leq R_k \qquad \forall k \in \mathcal{R}, \forall q \in Q \tag{15}$$

$$s_j^q = 0 \qquad \forall j \in \mathcal{J}, \forall q \in Q, t_{q+1} \leq r_j \wedge t_q \geq \tilde{d}_j \tag{16}$$

$$0 \leq s_j^q \leq s_j^{\max} \qquad \forall j \in \mathcal{J}, \forall q \in Q \tag{17}$$

Since $|\mathcal{T}| \leq 2|\mathcal{J}|$, the linear program has a polynomial number of variables and constraints, and can consequently be solved in polynomial time. Then, solving P-CTRTP-TW can be achieved by a binary search in polynomial time w.r.t. the required precision.

4.2 Relaxation-induced heuristics for the CTRTP-TW

In this section, we propose a preemptive relaxation-induced heuristic to turn a preemptive solution into a feasible non-preemptive one. Suppose that a feasible preemptive solution $(s_j^q)_{j \in \mathcal{J}}^{q \in Q}$ is available for a given P-CTRTP-TW(λ). Then several approached may be tried:

- Smooth the solution while keeping the same safety margin λ: In most cases such an approach will fail in satisfying the capacity constraints.
- Derive from the solution $(s_j^q)_{j \in \mathcal{J}}^{q \in Q}$ a sequencing π for the activities and then search for optimal evacuation rates by solving the CTRTP-TW(π) of Section 3.

4.2.1 Smoothing the processing rate while maintaining the safety margin

We propose in this context two relaxation-induced heuristics.

Heuristic RIH1 computes for each activity $j \in \mathcal{J}$ the time point indices $q_j^{\min} = \min_{q \in Q, s_j^q > 0} q$, $q_j^{\max} = \max_{q \in Q, s_j^q > 0} q$, and the sets the start and end times of the activities to $S_j = t_{q_j^{\min}}$ and $C_j = t_{q_j^{\max}+1}$. A consistent evacuation rate is then set to $s_j = P_j/(C_j - S_j)$. A drawback is that we may excessively increase the evacuation rates of the activities on some intervals, causing a violation of the resource constraints, especially for the first interval $[t_{q_j^{\min}}, t_{q_j^{\min}+1}]$ and the last one $[t_{q_j^{\max}}, t_{q_j^{\max}+1}]$.

To deal with this issue, heuristic RIH2 identifies the activities whose processing rate in the first interval and/or in the last interval is smaller than the mean value. Let $\overline{s}_j = P_j/(t_{q_j^{\max}+1} - t_{q_j^{\min}})$ denote the average processing rate value. The set of activities $\mathcal{J}^0$ is built with activities such that $q_j^{\max} \geq q_j^{\min} + 2$ (activities that cover at least three intervals) and that are such that $s_j^{q_j^{\min}} \leq \overline{s}$ and $s_j^{q_j^{\max}} \leq \overline{s}$. The set of activities $\mathcal{J}^1$ (resp. $\mathcal{J}^2$) is built with activities $j \notin \mathcal{J}^0$ such that $q_j^{\max} \geq q_j^{\min} + 1$ (activities that cover at least two intervals) and such that $s_j^{q_j^{\min}} \leq \overline{s}$ (resp. $s_j^{q_j^{\max}} \leq \overline{s}$).

Note that the three sets are disjoint and form a partition of the activities having at least one interval among the first one and the last one that has a rate lower than the mean value. An activity of $\mathcal{J}^0$ is excluded by construction from both $\mathcal{J}^1$ and $\mathcal{J}^2$. Sets $\mathcal{J}^1$ and $\mathcal{J}^2$ are also disjoint: if the activity covers more than three intervals then its rate cannot be lower than the mean both for the first and last interval, otherwise it would belong to in $\mathcal{J}^0$. Such an activity belongs by definition either to $\mathcal{J}^1$ or to $\mathcal{J}^2$, but not to both. If the activity covers only two intervals, then by definition its rate cannot be lower than the mean value both at the first and second interval.

The heuristic then assigns the following processing rates, start times and end times to the activities:

$$\forall j \in \mathcal{J}^0, \quad s_j = \frac{P_j - (t_{q_j^{\min}+1} - t_{q_j^{\min}}) s_j^{q_j^{\min}} - (t_{q_j^{\max}+1} - t_{q_j^{\max}}) s_j^{q_j^{\max}}}{t_{q_j^{\max}} - t_{q_j^{\min}+1}}$$

$$\forall j \in \mathcal{J}^1, \quad s_j = \frac{P_j - (t_{q_j^{\min}+1} - t_{q_j^{\min}}) s_j^{q_j^{\min}}}{t_{q_j^{\max}+1} - t_{q_j^{\min}+1}}$$

$$\forall j \in \mathcal{J}^2, \quad s_j = \frac{P_j - (t_{q_j^{\max}+1} - t_{q_j^{\max}}) s_j^{q_j^{\max}}}{t_{q_j^{\max}} - t_{q_j^{\min}}}$$

$$\forall j \in \mathcal{J}^0 \cup \mathcal{J}^1, \quad S_j = t_{q_j^{\min}+1} - \frac{(t_{q_j^{\min}+1} - t_{q_j^{\min}}) s_j^{q_j^{\min}}}{s_j}$$

$$\forall j \in \mathcal{J}^0 \cup \mathcal{J}^2, \quad C_j = t_{q_j^{\max}} + \frac{(t_{q_j^{\max}+1} - t_{q_j^{\max}}) s_j^{q_j^{\max}}}{s_j}$$

$$\forall j \in \mathcal{J}^1, \quad C_j = t_{q_j^{\max}+1}$$

$$\forall j \in \mathcal{J}^2, \quad S_j = t_{q_j^{\min}}$$

These assignments have the objective to avoid scheduling the activities at the beginning of their first interval and at the end of their last interval when they used less resource than the average consumption.

4.2.2 Derive sequencing first, schedule second

The values S_j and C_j computed for each activity $j \in \mathcal{J}$ by heuristic RIH1 can be used to derive a sequencing π. Then we solve the convex CTRTP-TW(π).

5 Flow-based Exact and heuristic methods

5.1 A flow-based branch-and-cut approach for the the CTRTP-TW

We propose a convex MINLP formulation based on the resource flow network model previously proposed for the RCPSP [7, 17]. This model considers resource units as flow units transferred from one activity to another and has already been used in case of continuous resources in [38].

Let $\Phi^k_{j_1,j_2}$ be a positive, continuous variable that gives the number of units of resource $k \in \mathcal{R}$ transferred from activity $j_1 \in \mathcal{J}_k$ to activity $j_2 \in \mathcal{J}_k$. We introduce additional dummy activities 0 and $|\mathcal{J}|+1$ that represent the start and end of the schedule, respectively. The flow-based formulation is then obtained by replacing the resource constraints (6) by the following ones:

$$\sum_{j \in \mathcal{J} \cup \{|\mathcal{J}|+1\}} \Phi^k_{0,j} = \sum_{j \in \mathcal{J} \cup \{0\}} \Phi^k_{j,|\mathcal{J}|+1} = R_k \qquad \forall k \in \mathcal{R} \tag{18}$$

$$\sum_{i \in (\mathcal{J}_k \cup \{0\}) \setminus \{j\}} \Phi^k_{i,j} = \sum_{i \in (\mathcal{J}_k \cup \{|\mathcal{J}|+1\}) \setminus \{j\}} \Phi^k_{j,i} = s_j \qquad \forall k \in \mathcal{R}, \forall j \in \mathcal{J}_k \tag{19}$$

$$\Phi^k_{i,j} > 0 \Rightarrow C_i \leq S_j \qquad \forall k \in \mathcal{R}, \forall i,j \in \mathcal{J}_k, i \neq j \tag{20}$$

Constraints (18) state that for each resource $k \in \mathcal{R}$ exactly R_k flow units are carried from the source node 0 to the sink node $|\mathcal{J}|+1$. Constraints (19) are the flow conservation constraints applied for each resource $k \in \mathcal{R}$ to each activity $j \in \mathcal{J}_k$. Finally, constraints (20) link the flow variables, the start and the end variables in such a way that activity i can only send flow units to an activity j that starts after the completion of i.

Note that the only nonlinear constraints, in the resulting formulation (7–12,18–20) are the convex duration constraints (8) and implication constraints (20).

A convex MINLP formulation is obtained by replacing the implication constraints (20) by the following constraint set [7], which require the introduction of a binary variable $x_{i,j}$, $\forall i,j \in \mathcal{J}, i \neq j$:

$$\Phi^k_{i,j} \leq \min(s^{\max}_i, s^{\max}_j) x_{i,j} \qquad \forall i,j \in \forall k \in \mathcal{R}, \forall i,j \in \mathcal{J}_k, i \neq j \tag{21}$$

$$S_j \geq C_i - (d_i - r_j)(1 - x_{i,j}) \qquad \forall i,j \in \mathcal{J}, i \neq j \tag{22}$$

$$x_{i,j} \in \{0,1\} \qquad \forall i,j \in \mathcal{J}, i \neq j \tag{23}$$

We opt for a branch-and-cut approach for solving this convex MINLP. Convex constraints (8) can be replaced by the following constraint set:

$$C_j - S_j \geq (-s_j + w)P_j/w^2 + P_j/w \quad \forall j \in \mathcal{J}, \forall w \in \mathbb{Q} \tag{24}$$

These constraints tell us that for any w, the 2D-point $(s_j, C_j - S_j)$ must be located above the tangent line in $(w, P_j/w)$ to the hyperbolic curve whose equation is $x \to P_j/x$. These constraints are difficult to handle since they are indexed on the rational numbers. We choose an integer K and we only keep among constraints (24) those which are related to a set of unformly chosen $K+1$ values $w = w_{j,k} = \left(k s^{\min}_j + (K-k) s^{\max}_j\right)/K$ for each activity $j \in \mathcal{J}$, and for each $k = 0, \ldots, K$. Then at each node of the branch-and-cut tree after Solving the LP relaxation, we search for j_0 and w_0 that violate (24). If such a pair cannot be found, the process stops. Otherwise the constraint related to j_0 and w_0 is inserted and the LP relaxation is solved again.

Figure 6 presents the network flow model of the CTRTP-TW solution of Fig. 1 with two differents flows, one for each resource.

5.2 An adaptive global insertion heuristic scheme

We now propose a heuristic that exploits the flow model. The main component of the heuristic is a procedure INSERT$(\lambda, \mathcal{L})$ that progressively inserts the activities in the flow network following a priority list $\mathcal{L}$ and trying to find a schedule with a safety margin of at least λ. The global procedure (Algorithm 1) takes as input a feasible $\lambda_{\min}$ (we set $\lambda_{\min} = 0$ in the experiments), an

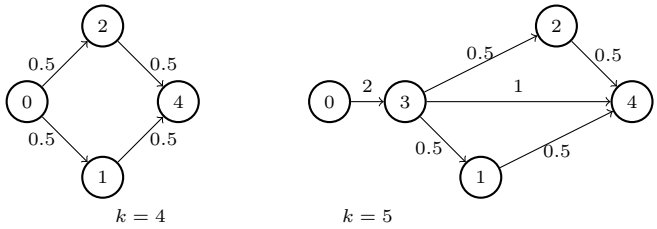


Figure 6: A network flow model of the CTRTP-TW solution of Fig. 1

upper bound $\lambda_{\max}$ on the safety margin (we use the preemtive upper bound), and searches the largest safety margin calling the INSERT procedure inside a binary search.

If INSERT is successful to find a schedule of at least λ, a sequencing π is derived from the obtained solution and a possibly improving solution is computed by solving the fixed sequencing convex problem CTRTP-TW(π), as explained in Section 3.

In case of failure, the INSERT procedure returns a list $\bar{\mathcal{L}}$ of forbidden ordered pairs of activities meaning that the next call of procedure Insert should change the relative order of the activities in each forbidden pair in the input list $\mathcal{L}$. To that purpose the forbidden pair list just returned by INSERT is added to the forbidden pair tabu-like list $\mathcal{F}$ and the UPDATE procedure searches for a new list compatible with the forbidden pairs in $\mathcal{F}$. If such a list is found, the INSERT procedure is called again. Otherwise (the forbidden pairs in $\mathcal{F}$ are inconsistent), a trial counter is incremented and the list is reinitialized randomly to diversify the search, while $\mathcal{F}$ is empied. The process stops when the desired precision is reached for the safety margin or if the trial counter reaches the maximal allowed value.

We explain how the list of activities $\mathcal{L}$ is initialized (Step 1). Intuitively, priority should be given to activities having both a small work content (i) and a small expected safety margin (ii): indeed (i) makes these activities easier to insert and (ii) means that these activities are more critical than the others and should be completed earlier. We define this expected safety margin as the quantity $\bar{\lambda}_j = d_j - \lambda - r_j - 2P_j/(s_j^{\max} + s_j^{\min})$. So we impose, for every generated list $\mathcal{L}$ that if j_1 and j_2 are such that $P_{j_1} < P_{j_2}$ and $\bar{\lambda}_{j_1} < \bar{\lambda}_{j_2}$ then $j_1 \prec_{\mathcal{L}} j_2$.

The INSERT procedure is precisey described and illustrated in Appendix A.

6 An exact dedicated branch-and-bound algorithm

Finally, we propose an exact Branch-and-Bound algorithm, which solves CTRTP-TW in an exact way, at least for small instances. This algorithm relies on the components that were presented during previous sections. Namely, the method embeds the optimistic estimation (upper bound) and the related relaxation-induced heuristics according to Section 4, and the flow-based insertion heuristic described in Section 5.2, all these methods making use of the fixed-sequencing solution procedure described in Section 3. What remains to be described is the nodes of related search tree and the branching scheme, the way the optimistic estimation of Section 4 is updated, the branching Strategy and the global tree search process.

Algorithm 1: The adaptive insertion heuristic

Input: a CTRTP-TW problem instance, an initial solution (S, C, s) of safety margin $\lambda_{\min}$, an upper bound $\lambda_{\max}$, a precision ϵ, a maximum number of trials $maxtrials$

Output: An improved solution (S, C, s) of safety margin λ

```
1  Initialize priority list L;
2  end ←false;
3  count ← 0;
4  while λ_max − λ_min ≥ ε and count < maxtrials do
5      λ ← (λ_min + λ_max)/2;
6      F ← ∅;
7      success ←false;
8      while ¬success and count < maxtrials do
9          (success, S, C, s, L̄) ← INSERT(λ, L);
10         if success then
11             λ_min ← λ;
12             count ← 0;
13             Derive a sequencing π from the solution and solve CTRTP-TW(π) to
                tentatively improve λ_min;
14         else
15             F ← F ∪ L̄;
16             if ¬UPDATE(L, F) then
17                 F ← ∅;
18                 count ← count + 1;
19                 Randomly generate a new list L;
```

6.1 Nodes of the search tree, branching scheme, and global search tree process

A node n of the search tree is a 5-tuple $\left((\hat{r}_j)_{j\in\mathcal{J}}, (\hat{d}_j)_{j\in\mathcal{J}}, \mathcal{M}, (\underline{m}_j)_{j\in\mathcal{M}}, (\overline{m}_j)_{j\in\mathcal{M}}\right)_n$, where $\hat{r}_j$ is the release date of activity $j \in \mathcal{J}$, $\hat{d}_j$ is the deadline of activity $j \in \mathcal{J}$, $\mathcal{M} \subseteq \mathcal{J}$ is the subset of activities j that are forced to have a mandatory part inside a "middle" interval $[\underline{m}_j, \overline{m}_j]$. The node elements must satisfy $\hat{r}_j < \hat{d}_j$, $\forall j \in \mathcal{J}$ and if $\mathcal{M} \neq \emptyset$, $\hat{r}_j \leq \underline{m}_j < \overline{m}_j \leq \hat{d}_j$, $\forall j \in \mathcal{M}$. The meaning of the middle interval for the concerned activities in set $\mathcal{M}$ is that to compute the preemptive relaxation associated with the node, we will force the variable processing rate $s_j(t)$ of each activity $j \in \mathcal{M}$ to be a constant if $t \in [\underline{m}_j, \overline{m}_j]$, to be increasing if $t \in [\hat{r}_j, \underline{m}]$, and to be decreasing if $t \in [\overline{m}_j, \hat{d}_j]$. Informally, the rate of each activity in $\mathcal{M}$ is constrained to be a quasiconcave step function of time.

Note that an activity $j \in \mathcal{M}$ such that $\hat{r}_j = \underline{m}_j$ and $\hat{d}_j = \overline{m}_j$ has a fixed start and completion time in solution space represented by the current node.

The branching scheme is based on an activity j^* and two time values α, β with $[\alpha, \beta] \subset [\hat{r}_{j^*}, \hat{d}_{j^*}]$. At most three children are generated from a given node:

- The "left" node, defined if $\hat{r}_{j^*} < \alpha$, is obtained by replacing the release date $\hat{r}_{j^*}$ by α: the task is constrained to start after α.
- The "middle" node, defined if $j^* \notin \mathcal{M}$ or if $j^* \in \mathcal{M}$, and $[\alpha, \beta] \supset [\underline{m}_{j^*}, \overline{m}_{j^*}]$ is obtained by inserting j^* in $\mathcal{M}$ (if not already present) and by setting or updating $\underline{m}_{j^*} \leftarrow \alpha$ and $\overline{m}_{j^*} \leftarrow \beta$: the task is constrained to be fully in process at a constant rate in $[\alpha, \beta]$.
- The "right" node, defined if $\hat{d}_{j^*} > \beta$, is obtained by replacing the deadline $\hat{d}_{j^*}$ by β: the task is constrained to end before β.

The union of the children solution spaces gives the father solution space and the solution space of each child node is strictly smaller than that of the father node. The triple (j^*, α, β) from which is issued the node is called the branching signature.

Now the global search tree is as follow. An initial lower bound is obtained by the flow-based heuristic presented in Section 5.2. At each node, an upper bound is computed via the preemptive relaxation, and relaxation-induced heuristics (both described in Section 4) are used to possibly update the upper bound. If the node cannot be pruned the branching scheme is applied by analysing the shape of the discrete rate function of the activities in the preemtpive schedule, with different stragegies depending on the concavity of the discrete rate function.

If there exists at least a non-quasiconcave activity l, i.e. the activity rate decreases after some time point t_{q_1} and increases after another time point t_{q_2} ($q_1 < q_2$) then a branching signature is created with $\alpha = t_{q_1+1}$, $\beta = t_{q_2}$, and $l \in \mathcal{M}$. If all activities have a quasiconcave step rate function, we distinguish between the activities that have at least three pieces in the rate function (type 1) and those that have only two pieces. For type 1 activities, we will set α and β to the start and end periods of the decreasing or the increasing part based on the highest decrease or increase. For type 2 activities, we just need one breakpoint α or β and we select one based on a rule aiming at smoothing the resource profile.

The upper-lower bounding process is described in Section 6.2. The branching strategy is detailed and illustrated in Appendix B. The completeness and finiteness proof is given in Appendix D.

6.2 Upper and lower bounds

An upper bound can be computed for each node by modifying the preemptive problem formulation P-CTRTP-TW(λ) described in Section 4.1 to take the contraints linked to the activities in $\mathcal{M}$ into account. Let λ denote a target safety margin. Let $\hat{\mathcal{T}} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$ denote the set of different release dates $\hat{r}_j$, deadlines $\hat{d}_j - \lambda$ for $j \in \mathcal{J}$, left bounds $\underline{m}_j$, right bounds $\overline{m}$ for $j \in \mathcal{M}$. Let $\hat{\mathcal{Q}}$ denote the interval index set $\hat{\mathcal{Q}} = \{1, \ldots, |\mathcal{T}| - 1\}$. The preemptive relaxation to check the feasibility of a given safety margin λ at each node n, P-CTRTP-TW(λ, n) is obtained from formulation (14–17), by replacing $\mathcal{Q}$ by $\hat{\mathcal{Q}}$ and adding the following constraints:

$$s_j^{q+1} \leq s_j^q \qquad \forall j \in \mathcal{M}, \forall q \in \hat{\mathcal{Q}} : t_q \geq \underline{m}_j \tag{25}$$

$$s_j^{q+1} \geq s_j^q \qquad \forall j \in \mathcal{M}, \forall q \in \hat{\mathcal{Q}} : t_{q+1} \leq \overline{m}_j \tag{26}$$

Note that these constraints enforce the increasing, then constant in $[\underline{m}_j, \overline{m}_j]$, then decreasing profile of s_j^q for all $j \in \mathcal{M}$.

Remark 1. *If $\mathcal{M} = \mathcal{J}$, $\hat{r}_j = \underline{m}_j$, $\hat{d}_j = \overline{m}_j$ for all $j \in \mathcal{M}$, then if P-CTRTP-TW(λ, n) admits a feasible solution, it is also feasible for CTRTP-TW(λ). Otherwise CTRTP-TW(λ) has no solution for node n.*

Now for a given node n, the upper bound is computed by a binary search to find the maximum preemptively feasible λ^* in $[\underline{\lambda}, \overline{\lambda}]$ where $\underline{\lambda}$ is the best known lower bound (feasible solution) and $\overline{\lambda}$ is the upper bound passed by the father node.

Note that each time a feasible preemptive solution is found in the binary search, the relaxation-induced procedures RIH1, RIH2, and RIH6 (see Section 4.2.1) are applied to possibly update the lower bound $\underline{\lambda}$ and the associated best known solution $(\underline{S}, \underline{C}, \underline{s})$. The flow-based convex program CTRTP-TW(π_n) can also be applied with the sequencing π_n induced by intervals $[\underline{m}_j, \overline{m}_j]$ for $j \in \mathcal{M}$.

As the upper bound optimization process depends on a required precision ϵ, during the generation of the linear program P-CTRTP-TW(λ, n), we merge any consecutive time points t_q, t_{q+1} such that $t_{q+1} - t_q < \epsilon$.

At the end of the binary search, we classically fall into one of these four possibilities:

- All linear programs P-CTRTP-TW(λ, n) where infeasible: no feasible preemptive solution exists and the node is pruned by infeasibility,
- The obtained upper bound λ^* (the best found preemptive solution) is such that $\lambda^* \leq \overline{\lambda}$. There is no need to further examine the node, which is pruned by bound.
- The obtained upper bound λ^* could be turned into a feasible solution with safety margin λ^* . The node is then pruned by local optimality.
- The obtained upper bound λ^* could not be turned into a feasible solution with the same safety margin and is such that $\lambda^* > \overline{\lambda}$. In this case the node cannot be pruned. We update $\overline{\lambda} \leftarrow \lambda^*$ and resort to branching.

7 Numerical Experiments

We ran the numerical experiments on a Linux (CentOS 7) Server with processors Intel Xeon CPU E7-8890 v3@2.50 GHz. The algorithms were implemented in C++ and compiled with gcc 7.3. The (mixed integer) linear models were solved using Cplex 12.8 in a single-thread mode with all its parameters set to their default values. The time limit was fixed to 1 hour. All software, data and results are available for download [5].

7.1 Instance generation and experimental setup

A part of the instances come from the GEOSAFE project and an evacuation network simulator (see [3]). They are related to the tree late evacuation problem (T-LEP), which gave rise to our general CTRTP-TW model. In this problem the evacuation network is a tree whose leaves are the places to be evacuated, the root is the safe node where evacuees must be regrouped, and the arcs have capacities corresponding to resource availabilities and lengths, which define the activity processing times. We refer to [16] and [3] to obtain the transformation from T-LEP to CTRTP-TW. The instances are clustered into 10 instance groups dense_x, medium_x, sparse_x, where x is the number of evacuee groups (corresponding to the number of activities), and dense, medium and sparse are related to the mean degree of the nodes in the related tree and an estimation of the congestion level on the arc-resources. Those instances are most often tight, in the sense that computing a feasible solution is difficult and in some cases impossible. We discard instances that have been checked not to admit any feasible solution.

For any group of 10 instances, we compute several indicators:

- A priori indicators:
  - The number of *nodes* of the related tree;
  - The minimal duration *capRelax* of the evacuation process in case capacity constraints are relaxed;
  - The mean (over all nodes x) ratio *congest*, between the sum of the capacities of the arc-resources that enter into x and the capacity of the arc-resource emanating from x.
- A posteriori indicators that express, for any instance, the characteristics of its optimal solution:
  - The ratio *compress* (weighted average for all P_j between maximal evacuation rate s_j^{max} and real rate s_j according to this optimal solution: a compression of the rate, which can also be seen as a congestion measure);
  - The global duration *duration* of the process.
  - The average number *paralAv* of activities that are simultaneously performed at some time t.
  - The maximal number *paralMax* of activities that are simultaneously performed at some time t.

	A priori indicators			A posteriori indicators			
Instances	**nodes**	**capRelax**	**congest**	**compress**	**duration**	**paralAv**	**paralMax**
dense_10	19.8	155.06	1.69	4.14	311.27	3.2	5.7
dense_15	29.1	160.08	1.78	4.24	409.78	4	6.9
dense_20 (9)	38.6	164.88	1.84	8.10	453.36	5.57	9.44
medium_10	19.7	152.83	1.71	6.33	339.96	3.72	5.8
medium_15	29.1	159.39	1.8	5.9	401.75	4.31	7
medium_20 (9)	38.2	160.69	1.86	6.75	474.96	5.37	9.33
medium_25	46.8	169.91	1.91	7.18	476.32	7.53	12
sparse_10	19.5	146.17	1.75	4.88	289.82	3.93	6.3
sparse_15	28.8	153.92	1.87	6.42	373.13	5.15	8.7
sparse_20	38.3	157.78	1.87	7.3	446.69	5.26	9.9
sparse_25	47.6	154.73	1.89	7.15	473.83	7.14	11.5

Table 2: T-LEP Instances characteristics

As the CTRTP-TW instances issued from T-LEP instances have a particular structure, we also generated general CTRTP-TW instances. We generated families of instances of 10 instances each. Each family is denoted `gen`X`_`a`_`b with $X = \{1, 2\}$ the resource tightness factor, $a \in \{15, 20, 25\}$ the number of activities, $b \in \{2, 5, 10\}$ the number of resources.

For each instance, the release dates are generated uniformly between 0 and 200. For each resource k, its capacity is uniformly generated between 50 and 100. For each resource k, each activity is added to $\mathcal{R}_k$ with probability 0.4. For each activity i, the maximum rate $s_i^{\max}$ is set to the minimum capacity of the resources it requires. The total work content P_i is uniformly generated between 1000 and 4000.

To obtain feasible instances, we build an auxiliary instance identical to the main instance, except that each activity i has now a fixed rate that depends on the resource tightness factor X. If $X = 1$, the fixed rate of an activity i in the auxiliary instance is equal to $\frac{s_i^{\max}}{4}$. If $X = 2$, the rate is fixed to $\min_{k \in \mathcal{R}_i} \frac{2}{|\mathcal{J}_k|} R_k$. The auxiliary activity duration is set to $\frac{P_i}{s_i}$. Then a feasible schedule for the auxiliary instance is obtained by the serial schedule generation scheme [25] taking the activities in the lexicographic order. The deadline of the activity in the main instance is set to its completion time in this solution. This way, the problem is feasible. The lower the resource factor tightness, the higher the resource consumption of the auxiliary task. So the time windows of the activities are tighter, which induces less possibilities for a parallel execution of the activities.

In the following sections, we compare the results obtained by the different methods with different parameters and we give average results for each group of instances studied.

The average is taken on the 10 instances of each group except when the compared methods do not provide a solution for all the instances of the group. In this case, the average is calculated only on the instances for which the compared methods have all given a solution.

7.2 Results

We first evaluate the quality of the preemptive bound, w.r.t the best lower bound found among all methods. Table 3 presents, for each instance family, the number of instances for which a feasible solution could be found by at least one evaluated method (column `#feas`), the number of instances for which the optimal solution is known (column `#opt`) and the number of instances for which the preemptive bound is tight (column `opt-p`). For all instances for which a feasible

solution is found, the table displays the average gap above the best found lower bound (`av gap LB*`), the avarage gap below the best found upper bound (`av gap UB*`), and the average cpu time. The table displays also the average gap below the optimal safety margin when the optimum is known (column `av gap opt`).

Instances	#feas	#opt	#opt-p	av gap LB*	av gap opt	av gap UB*	cpu (s)
dense_10	10	9	9	0.20%	0.00%	-0.21%	0.028
dense_15	10	9	7	3.99%	2.92%	-3.93%	0.052
dense_20	9	8	8	4.39%	0.00%	0.00%	0.077
medium_10	10	9	9	0.64%	0.00%	-0.68%	0.024
medium_15	10	7	6	1.64%	0.82%	-1.28%	0.052
medium_20	9	6	5	3.84%	1.77%	-1.22%	0.080
medium_25	10	9	9	6.65%	0.00%	0.00%	0.101
sparse_10	10	10	9	0.02%	0.02%	-0.02%	0.028
sparse_15	10	9	9	0.00%	0.00%	-0.01%	0.040
sparse_20	10	6	6	1.96%	0.00%	-0.16%	0.078
sparse_25	10	7	7	3.24%	0.00%	0.00%	0.103
gen1_15_2	10	9	6	4.81%	2.73%	-4.72%	0.041
gen1_15_5	10	10	7	6.51%	6.51%	-6.52%	0.043
gen1_15_10	10	10	5	6.63%	6.63%	-6.63%	0.049
gen1_20_2	10	7	4	7.94%	3.68%	-3.00%	0.074
gen1_20_5	10	8	6	5.28%	3.04%	-4.76%	0.061
gen1_20_10	10	5	2	10.18%	9.61%	-6.84%	0.075
gen1_25_2	10	8	8	1.43%	0.00%	0.00%	0.082
gen1_25_5	10	7	7	3.18%	0.00%	-0.29%	0.095
gen1_25_10	10	3	2	19.49%	6.08%	-3.07%	0.105
gen2_15_2	10	10	6	5.92%	5.92%	-5.92%	0.040
gen2_15_5	10	10	7	5.82%	5.82%	-5.82%	0.042
gen2_15_10	10	10	5	19.06%	19.06%	-19.06%	0.051
gen2_20_2	10	10	8	1.36%	1.36%	-1.36%	0.063
gen2_20_5	10	10	5	7.35%	7.35%	-7.35%	0.071
gen2_20_10	10	8	5	3.18%	1.78%	-2.03%	0.081
gen2_25_2	10	9	9	0.13%	0.00%	0.00%	0.093
gen2_25_5	10	9	8	1.69%	1.67%	-1.50%	0.114
gen2_25_10	10	6	4	3.65%	3.49%	-2.23%	0.126

Table 3: Preemptive bound evaluation

Column `#opt` gives an indication about the instance difficulty for the tested methods. Clearly the number of known optima decreases with the number of activities for the evacuation instances and with the number of activities and resources for the general instances. Furthermore, the gen1 instances are more difficult than the gen2 instances. The quality of the preemtive bound is rather good for the evacuation instances (less than 7% from the best LB for the worst average). For the general instances the distance from the best UB increases with the number of resources (except for instances gen2_20).

We now evaluate the adaptive insertion heuristic. The results are displayed in Table 4. Columns `#feas` and `#opt` give the number of times the heuristic found a feasible (resp. optimal) solution. As for the preceding table, the other columns give the average gap below the best known solution, below the optimal solution when it is known, and below the best known upper bound for each instance family.

Instances	#feas	#opt	av gap LB*	av gap opt	av gap UB*	cpu (s)
dense_10	10	7	-0.28%	-0.18%	0.27%	2.37
dense_15	10	8	-1.69%	-0.30%	1.75%	14.60
dense_20	8	7	-2.63%	-2.63%	2.63%	39.64
medium_10	10	8	-0.59%	-0.04%	0.55%	1.90
medium_15	10	5	-4.24%	-2.02%	4.57%	17.28
medium_20	8	6	-6.79%	0.00%	7.31%	76.44
medium_25	9	7	-3.74%	-3.74%	3.74%	95.73
sparse_10	10	9	-0.02%	-0.02%	0.02%	3.04
sparse_15	10	9	-0.69%	0.00%	0.68%	4.67
sparse_20	10	6	-4.99%	0.00%	6.63%	33.90
sparse_25	9	5	-8.76%	-2.12%	9.80%	137.73
gen1_15_2	10	6	-4.63%	-3.49%	4.73%	2.53
gen1_15_5	10	6	-2.58%	-2.58%	2.58%	4.54
gen1_15_10	10	4	-11.19%	-11.19%	11.19%	8.22
gen1_20_2	10	3	-22.21%	-10.55%	24.21%	7.68
gen1_20_5	10	2	-26.93%	-22.69%	27.34%	12.16
gen1_20_10	10	1	-30.95%	-29.58%	32.71%	23.01
gen1_25_2	10	2	-16.13%	-18.65%	17.48%	9.76
gen1_25_5	9	1	-26.12%	-19.83%	26.64%	21.24
gen1_25_10	8	0	-41.11%	-29.75%	45.22%	34.24
gen2_15_2	10	6	-4.63%	-3.49%	4.73%	2.53
gen2_15_5	10	6	-9.76%	-9.76%	9.75%	5.23
gen2_15_10	10	3	-10.19%	-10.19%	10.19%	8.43
gen2_20_2	10	7	-3.25%	-3.25%	3.25%	2.44
gen2_20_5	10	5	-18.30%	-18.30%	18.30%	9.89
gen2_20_10	10	3	-20.16%	-17.43%	21.06%	18.12
gen2_25_2	10	7	-1.83%	-0.89%	1.94%	5.50
gen2_25_5	10	6	-9.95%	-7.91%	10.08%	11.70
gen2_25_10	10	3	-15.03%	-4.74%	16.08%	31.25

Table 4: Adaptive insertion heuristic evaluation

For the evacuation instances, the gap from the best known solution increases with the instance size. However the gap from the optimum is remarkably low when the optimum is known. For the general instances, the gaps are much higher and they systematically increase with the number of resources. The gaps are also much higher for the gen1 instances than for the gen2 instances. A possible explanation for this behavior is that the gen2 instances allow more parallelism among activity execution, which is better exploited by the heuristic. To support this hypothesis, we display in Figure 7, the schedule obtained by the heuristic for one of the gen2 instances (at the top right of the figure) and for a gen1 instance (at the bottom right of the figure). The figure displays at the top left the fixed-rate schedule that was used to generate the gen2 instance, while et the bottom left, the fixed-rate schedule that we used to generate the gen1 instance is displayed. One observes much more parallelism in the gen2 schedules than in the gen1 schedules. Furthermore, there is also a high level of parallelism allowed in the evacuation instances because the deadlines are initially assigned to the resources (arcs of the evacuation network see [2, 16]) where all the evacuated population converge.

We provide the results of the branch-and-cut method for each instance family in Table 5. Column `#feas` gives the number of times the branch-and-cut issued a feasible solution. Column `opt` provides the number of optimal solutions found and proved to be optimal. Column `av gap`

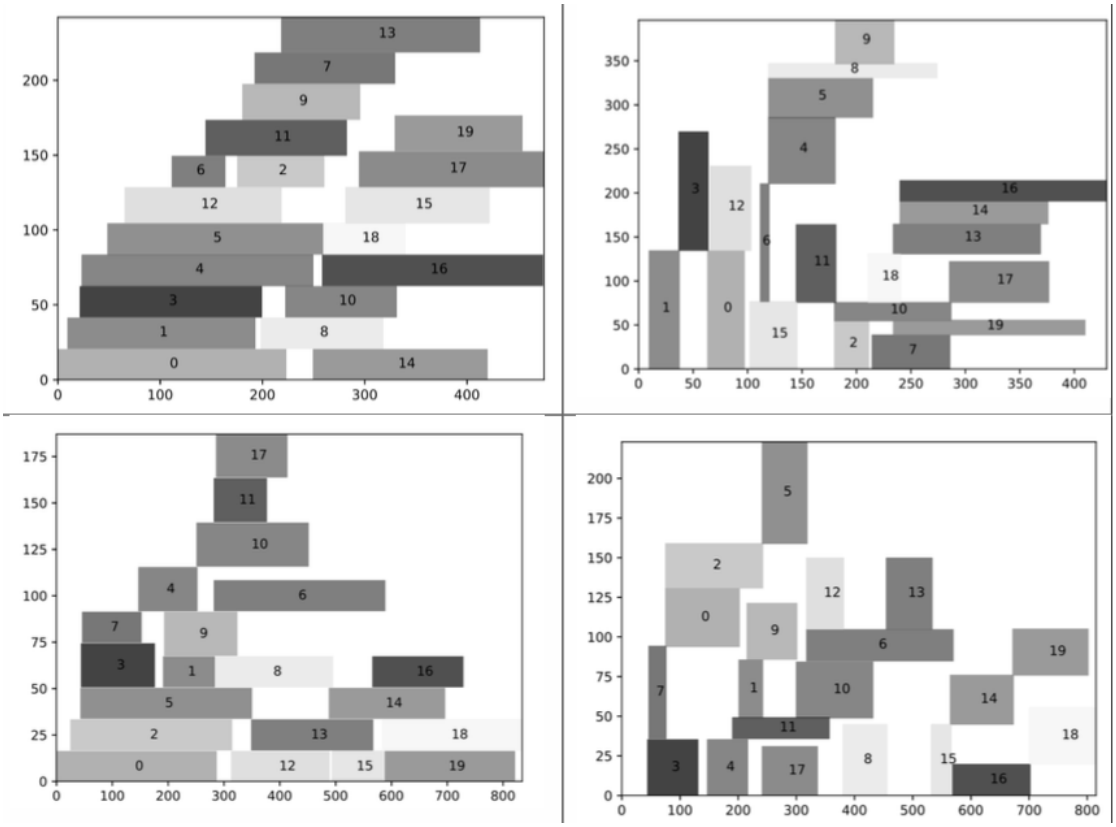


Figure 7: Comparing generation and solution for gen2 (top) and gen1 (bottom) instances

gives the average gap at the end of the branch-and-bound process. Column `av gap LB*` gives the average gap between the obtained LB and the best LB found by all methods. Column `av gap UB*` displays the average gap beween the best UB in the branch-and-bound tree and the the best UB found by all methods. Column `#cuts` gives the average number of generated cuts (24), column `#nodes` gives the average total number of explored nodes, and column `#cpu` gives the average cpu time.

Instances	#feas	#opt	av gap	av gap LB*	av gap UB*	#cuts	#nodes	cpu (s)
dense_10	10	10	0.00%	**0.00%**	0.00%	210.9	20479.7	49.6
dense_15	10	9	1.54%	**-0.01%**	**1.45%**	508.4	62867.4	392.0
dense_20	7	6	10.96%	**0.00%**	10.96%	802.2	322689.7	1932.1
medium_10	10	10	0.00%	**0.00%**	0.04%	218.3	106121.4	117.9
medium_15	10	6	20.80%	**-0.10%**	20.00%	825.2	416296	1462.8
medium_20	8	6	30.34%	**0.00%**	28.27%	836.7	206646.7	1522.4
medium_25	9	7	5.15%	**-0.18%**	4.98%	1647.6	114938	1890.9
sparse_10	10	10	0.00%	**0.00%**	**0.00%**	170.7	1360	5.2
sparse_15	10	10	0.00%	**0.00%**	**0.00%**	331.7	22610.1	98.8
sparse_20	10	6	7.38%	**0.00%**	5.21%	1499.4	144521.9	1476.5
sparse_25	9	4	61.62%	**0.00%**	57.16%	2276.2	158238.3	2512.9
gen1_15_2	10	9	4.46%	**0.00%**	4.28%	293	341379	360.1
gen1_15_5	10	10	0.00%	**0.00%**	**0.00%**	256.4	1003.9	1.0
gen1_15_10	10	10	0.00%	**0.00%**	**0.00%**	358.9	31335.4	40.9
gen1_20_2	10	6	8.14%	**0.00%**	**1.41%**	1294.9	925538	1456.9
gen1_20_5	10	8	0.56%	**0.00%**	**0.00%**	992.3	381065	1090.6
gen1_20_10	10	3	6.24%	**-0.06%**	**2.04%**	1388	413470	3197.5
gen1_25_2	10	7	2.02%	**-0.46%**	**0.00%**	2097	747578	2191.4
gen1_25_5	9	7	0.87%	**0.00%**	**0.00%**	2160.9	351608.4	2524.2
gen1_25_10	9	3	81.51%	-10.76%	**1.26%**	3231.7	195242.6	3245.0
gen2_15_2	10	10	0.00%	**0.00%**	**0.00%**	252.3	3472.9	4.0
gen2_15_5	10	10	0.00%	**0.00%**	**0.00%**	208.6	470.4	0.4
gen2_15_10	10	10	0.00%	**0.00%**	**0.00%**	515.2	5114.7	7.9
gen2_20_2	10	10	0.00%	**0.00%**	**0.00%**	914.3	96602.8	217.7
gen2_20_5	10	10	0.00%	**0.00%**	**0.00%**	1133	73190	302.5
gen2_20_10	10	8	1.72%	**0.00%**	**0.60%**	1081.9	201398	1608.4
gen2_25_2	10	7	1.76%	**-0.02%**	1.77%	1446.8	374876	1118.6
gen2_25_5	10	8	0.84%	**-0.66%**	**0.00%**	2526.8	235104	1618.8
gen2_25_10	10	5	4.58%	-3.39%	**0.00%**	2035.1	125233	1957.3

Table 5: Flow-based branch-and-cut results

Results displayed in bold indicate that the method obtained the best average lower or upper bound. The result show that in terms of best solutions found, the branch-and-cut method performs remarkably well compared to the other tested methods for both the general and evacuation instances. An exception is for the generalized instances with 25 activities and 10 resources, where the discrete model finds on average better solutions. The branch-and-cut experiences difficulties in closing the evacuation and gen1 instances, for which large gaps can be seen. However, the method obtains the best upper bounds for almost all generalized instances.

The results of the dedicated branch-and-bound method are displayed in Table 6. The table displays the same indicators as for the branch-and-cut method, except for the `#cuts` column.

Instances	#feas	#opt	av gap	av gap LB*	av gap UB*	#nodes	cpu (s)
dense_10	10	10	0.00%	**0.00%**	**0.00%**	6525	150
dense_15	10	7	2.99%	**-0.01%**	2.92%	28983	1089
dense_20	8	8	0.00%	**0.00%**	**0.00%**	10385	726
medium_10	10	10	0.00%	-0.04%	**0.00%**	8529	269
medium_15	10	6	1.59%	-0.65%	**0.53%**	28139	1441
medium_20	8	5	8.41%	-4.43%	**0.94%**	30211	1800
medium_25	9	7	1.59%	-1.48%	**0.00%**	10420	1080
sparse_10	10	9	0.02%	**0.00%**	0.02%	41509	360
sparse_15	10	9	0.00%	**0.00%**	0.01%	12331	360
sparse_20	10	6	4.02%	-1.64%	**0.00%**	16689	1440
sparse_25	9	6	9.14%	-5.28%	**0.00%**	12220	1440
gen1_15_2	10	7	2.75%	-2.10%	**0.28%**	55548	1470
gen1_15_5	10	6	6.64%	-1.23%	5.30%	42615	1449
gen1_15_10	10	6	8.28%	-4.32%	2.92%	28392	1477
gen1_20_2	10	3	29.51%	-10.51%	2.10%	52563	2521
gen1_20_5	10	3	19.91%	-9.99%	4.48%	53809	2677
gen1_20_10	10	1	23.00%	-9.60%	5.80%	40250	3240
gen1_25_2	10	2	14.74%	-10.96%	**0.00%**	47223	2880
gen1_25_5	10	2	67.77%	-24.61%	0.30%	43057	2880
gen1_25_10	10	1	77.17%	-22.42%	2.28%	23920	3545
gen2_15_2	10	8	4.20%	-3.70%	0.64%	28693	1038
gen2_15_5	10	8	2.85%	-0.39%	3.27%	49250	1440
gen2_15_10	10	4	11.85%	-1.26%	14.44%	35708	2198
gen2_20_2	10	8	3.20%	-2.52%	0.86%	17081	721
gen2_20_5	10	5	11.78%	-5.78%	9.18%	33102	1804
gen2_20_10	10	3	9.34%	-7.08%	1.35%	37447	2520
gen2_25_2	10	7	1.59%	-1.48%	**0.00%**	15717	1080
gen2_25_5	10	6	7.76%	-6.43%	1.77%	10401	1440
gen2_25_10	10	4	15.13%	-12.21%	2.38%	15809	2200

Table 6: Branch-and-bound results

In terms of best lower bounds, the branch-and-bound method is generally outperformed by the branch-and-cut method except for about half of the evacuation instances. On these intances, the branch-and-bound method finds better upper bounds than the branch-and-cut method. The branch-and-bound method appears to be sensitive to the quality of the preemtive bound and of the initialization heuristic. When both values are close to the optimum, as it happens on evacuation instances, only a few nodes are explored and the branch-and-bound quickly converges towards the optimum. On the general instances, both the preemptive bound and the insertion heuristic have larger optimality gaps. In this case, the gap stays rather high and the time limit is reached. An important element is the precision ϵ, which was fixed here to $\epsilon = 0.001$. We observed empirically that changing the precision has little influence on the CPU time. The best upper bound is also stable. However the lower bound is more sensitive to the precision change (see Appendix C).

Finally, we compare the continuous and discrete model by solving the DTRTD-TW model detailed in [16] with the CP Optimizer solver, under a 3600s time limit. The results are displayed in Table 7.

Instances	#feas	av gap LB*	cpu (s)
dense_10	10	-1.27%	0.57
dense_15	10	-2.80%	3.09
dense_20	9	-8.20%	719.38
medium_10	10	-0.83%	0.13
medium_15	10	-1.84%	364.92
medium_20	9	-1.15%	1097.35
medium_25	9	-1.68%	717.75
sparse_10	10	-0.76%	0.02
sparse_15	10	-1.39%	0.08
sparse_20	10	-1.55%	376.56
sparse_25	10	-4.81%	1437.12
gen1_15_2	10	-3.07%	0.03
gen1_15_5	10	-2.45%	0.10
gen1_15_10	10	-2.89%	0.13
gen1_20_2	10	-3.12%	0.16
gen1_20_5	10	-3.63%	0.36
gen1_20_10	10	-2.93%	4.05
gen1_25_2	10	-3.21%	0.28
gen1_25_5	10	-2.85%	80.80
gen1_25_10	10	**-4.16%**	374.26
gen2_15_2	10	-3.05%	0.01
gen2_15_5	10	-3.49%	0.07
gen2_15_10	10	-4.97%	0.19
gen2_20_2	10	-1.30%	0.05
gen2_20_5	10	-2.76%	0.19
gen2_20_10	10	-1.88%	0.94
gen2_25_2	10	-1.26%	0.16
gen2_25_5	10	-1.63%	2.01
gen2_25_10	10	**-1.35%**	3.09

Table 7: Discrete model results

The results show that smaller satefy margins are obtained due to discretization, sometimes significantly. Evacuation instances are challenging even in the discrete setting, as the time limit is reached several times without proving optimality. However, solving the discrete model is a good heuristic, as it even gets the best average gaps from the best known lower bound for the two largest generalized instances.

8 Conclusion

We have introduced a variant of the cumulative scheduling problem (CuSP) with Safety Margin criterion, which involves both discrete features and continuous ones. This model can be used to schedule evacuation processes in case of natural disaster. We have provided a full complexity map of the problem and showed that it is difficult to handle in the general case. We have formulated the fixed-sequencing variant of the problem as a convex program, from which we derive a branch-and-cut method to solve the general problem. We have studied the preemptive version of the problem, which can be solved by linear programming and provides upper bounds. A network flow-based heuristic has been proposed to obtain a lower bound. An exact branch-and-bound algorithm for the general problem uses these lower and upper bounding components as well as

an original branching scheme. The computational experiments have shown the merits and the drawbacks of the dedicated branch-and-bound method compared to the flow-based branch-and-cut algorithm on evacuation-based and general instances. They have also shown the interest of the continuous model compared to the discrete approximation proposed in [16]. Further work will aim at:

- Enhancing the filtering ability of the branch-and-bound algorithm, through constraint propagation and the use of a better fitted branching mechanism;
- Extending the model to precedence constraints;
- Studying the way the model may be applied to potential industrial applications or evacuation issues in the case when the evacuation network is not a tree.

Acknowledgements

This work has been carried on in the context of the H2020 Marie Skłodowska-Curie Research and Innovation Staff Exchange European project 691161 "GEO-SAFE" [19]. This work has also been supported by ANITI, the Artificial and Natural Intelligence Toulouse Institute.

The authors would like to warmly thank Alicia Vanhulle for her help in the computational tests of the CP model.

References

[1] Christian Artigues, Sophie Demassey, and Emmanuel Neron, editors. *Resource-constrained project scheduling*. Wiley Online Library, 2008.

[2] Christian Artigues, Emmanuel Hébrard, Yannick Pencolé, Andreas Schutt, and Peter J Stuckey. A Study of Evacuation Planning for Wildfires. In *The Seventeenth International Workshop on Constraint Modelling and Reformulation (ModRef 2018)*, Lille, France, 2018.

[3] Christian Artigues, Emmanuel Hébrard, Yannick Pencolé, Andreas Schutt, and Peter J. Stuckey. Data instance generator and optimization models for evacuation planning in the event of wildfire. In *Proceedings of the GEOSAFE Workshop on Robust Solutions for Fire Fighting, RSFF 2018, L'Aquila, Italy, July 19-20, 2018.*, pages 75–86, 2018.

[4] Christian Artigues, Emmanuel Hébrard, Alain Quilliot, and Hélène Toussaint. Multi-mode RCPSP with safety margin maximization: Models and algorithms. In *10th International Conference on Operations Research and Enterprise Systems, ICORES*, pages 129–136, 2021.

[5] Christian Artigues, Emmanuel Hébrard, Alain Quilliot, and Hélène Toussaint. The continuous time-resource tradeoff scheduling problem with time windows, 2024. Available for download at https://github.com/INFORMSJoC/2022.0142.

[6] Christian Artigues and Pierre Lopez. Energetic reasoning for energy-constrained scheduling with a continuous resource. *Journal of Scheduling*, 18(3):225–241, 2015.

[7] Christian Artigues, Philippe Michelon, and Stéphane Reusser. Insertion techniques for static and dynamic resource constrained project scheduling. *European Journal of Operational Research*, 149(2):249–267, 2003.

[8] Vedat Bayram. Optimization models for large scale network evacuation planning and management: A literature review. *Surveys in Operations Research and Management Science*, 21(2):63–84, 2016.

[9] Umut Beşikci, Ümit Bilge, and Gündüz Ulusoy. Multi-mode resource constrained multi-project scheduling and resource portfolio problem. *European Journal of Operational Research*, 240(1):22–31, 2015.

[10] Peter Brucker and Jürgen Neyer. Tabu-search for the multi-mode job-shop problem. *OR Spectrum*, 20(1):21–28, 1998.

[11] Jacques Carlier, Abderrahim Sahli, Antoine Jouglet, and Eric Pinson. A faster checker of the energetic reasoning for the cumulative scheduling problem. *International Journal of Production Research*, 60(11):3419–3434, 2022.

[12] Stéphane Dauzère-Pérès, William Roux, and Jean-Bernard Lasserre. Multi-resource shop scheduling with resource flexibility. *European Journal of Operational Research*, 107(2):289–305, 1998.

[13] Erik Demeulemeester, Bert De reyck, and Willy Herroelen. The discrete time/resource trade-off problem in project networks: a branch-and-bound approach. *IIE Transactions*, 32(11):1059–1069, 2000.

[14] Erik Demeulemeester and Willy S Herroelen. *Project scheduling: a research handbook*, volume 49. Springer Science & Business Media, 2006.

[15] Caroline Even, Victor Pillac, and Pascal Van Hentenryck. Convergent plans for large-scale evacuations. In *AAAI*, pages 1121–1127, 2015.

[16] Caroline Even, Andreas Schutt, and Pascal Van Hentenryck. A constraint programming approach for non-preemptive evacuation scheduling. In *International Conference on Principles and Practice of Constraint Programming*, pages 574–591. Springer, 2015.

[17] Philippe Fortemps and Maciej Hapke. On the disjunctive graph for project scheduling. *Foundations of Computing and Decision Sciences*, 22(3):195–209, 1997.

[18] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[19] GEO-SAFE. *Geospatial based Environment for Optimisation Systems Addressing Fire Emergencies*, 2020. `https://cordis.europa.eu/project/id/691161` [Accessed: 21/12/2020].

[20] Michel Habib, Ross McConnell, Christophe Paul, and Laurent Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1-2):59–84, 2000.

[21] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010.

[22] Mohd Hafiz Hasan and Pascal Van Hentenryck. Large-scale zone-based evacuation planning–part i: Models and algorithms. *Networks*, 77:127–146, 2020.

[23] Willy Herroelen. Project scheduling—theory and practice. *Production and Operations Management*, 14(4):413–432, 2005.

[24] W. A. Horn. Some simple scheduling algorithms. *Naval Research Logistics Quarterly*, 21(1):177–185, 1974.

[25] Rainer Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333, 1996.

[26] Rainer Kolisch. *Project scheduling under resource constraints: efficient heuristics for several problem classes*. Springer Science & Business Media, 2013.

[27] Anulark Naber. Resource-constrained project scheduling with flexible resource profiles in continuous time. *Computers & Operations Research*, 84:33–45, 2017.

[28] Anulark Naber and Rainer Kolisch. Mip models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, 239(2):335–348, 2014.

[29] Margaux Nattaf, Christian Artigues, and Pierre Lopez. A hybrid exact method for a scheduling problem with a continuous resource and energy constraints. *Constraints*, 20(3):304–324, 2015.

[30] Margaux Nattaf, Markó Horváth, Tamás Kis, Christian Artigues, and Pierre Lopez. Polyhedral results and valid inequalities for the continuous energy-constrained scheduling problem. *Discrete Applied Mathematics*, 258:188–203, 2019.

[31] Alain Quilliot, Christian Artigues, Emmanuel Hebrard, and Hélène Toussaint. Models and algorithms for natural disaster evacuation problems. In *Proceedings of the 2019 Federated Conference on Computer Science and Information Systems, FedCSIS 2019, Leipzig, Germany, September 1-4, 2019.*, pages 143–146, 2019.

[32] Christoph Schwindt and Jürgen Zimmermann, editors. *Handbook on Project Management and Scheduling*, volume 1. Springer, 2015.

[33] S Mohammad Seyed-Hosseini and Majid Sabzehparvar. A mathematical model for the continuous time/resource trade-off problem. *Kuwait Journal of Science and Engineering*, 35(1B):197–215, 2008.

[34] Shahrooz Shahparvari, Prem Chhetri, Babak Abbasi, and Ahmad Abareshi. Enhancing emergency evacuation response of late evacuees: Revisiting the case of australian black saturday bushfire. *Transportation research part E: Logistics and Transportation Review*, 93:148–176, 2016.

[35] Daniel W Steeneck and Subhash C Sarin. Resource-constrained project scheduling with concave processing rate functions. *Journal of the Operational Research Society*, 66(5):794–806, 2015.

[36] Sha Tao and Zhijie Sasha Dong. Multi-mode resource-constrained project scheduling problem with alternative project structures. *Computers & Industrial Engineering*, 125:333–347, 2018.

[37] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006.

[38] Martin Tritschler, Anulark Naber, and Rainer Kolisch. A hybrid metaheuristic for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, 262(1):262–273, 2017.

[39] Vincent Van Peteghem and Mario Vanhoucke. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, 235(1):62–72, 2014.

[40] Grzegorz Waligóra. Heuristic approaches to discrete-continuous project scheduling problems to minimize the makespan. *Computational Optimization and Applications*, 48(2):399–421, 2011.

[41] Jan Węglarz, Joanna Józefowska, Marek Mika, and Grzegorz Waligóra. Project scheduling with finite or infinite number of activity processing modes–a survey. *European Journal of Operational Research*, 208(3):177–205, 2011.

[42] Moli Yang, Andreas Schutt, and Peter J Stuckey. Time table edge finding with energy variables. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 633–642. Springer, 2019.

A The INSERT procedure

We describe the main INSERT procedure. The procedure takes the activities in the order defined by list $\mathcal{L}$ and inserts them one by one in the flow to build the schedule. Algorithm 2 describes the procedure. A list of already scheduled activities is initialized to the dummy start activity 0 (Step 1) and the flow values are all set to 0 (Step 2). The main loop (Steps 3–32) scans all the activities j in the input list $\mathcal{L}$. Let $\bar{\Phi}^k_{i,j}$ denote the maximum amount of flow that can be sent from an already inserted activity i to the activity j to be inserted. On each resource k, Steps 5–6 set this flow to the minimum between the maximum rate of activity j and the part of s_i not already sent to another activity of $\mathcal{Q}$, with the particular case of $i = 0$ having a maximum number of units to send equal to R_k.

Then, for each resource k, in loop 9–15, the already scheduled activities of $\mathcal{Q}$ are scanned in the increasing order of their completion times as candidates for sending the flow units required by j. Because all resources are treated separately at this stage, the start time and the number of units may differ on each resource required by j and are consequently stored in auxiliary variables S^k_j and s^k_j, respectively. For each such activity $i \in \mathcal{Q} \cap \mathcal{J}_k$, the maximum amount of flow received by j considering the additional units sent by i is stored in *auxrate* (Step 10). Because of the ordering of set $\mathcal{Q}$, the earliest start time on resource k for activity j if it receives some units of flow sent from i as well as from predecessors of i in $\mathcal{Q}$ is equal to $\max(r_j, C_i)$. If *auxrate* satisfies the minimum rate requirement of activity j and if the start time plus the duration obtained with *auxrate* does not exceed the safety margin (condition checked at Step 11), then a feasible insertion has been found. The start time is set and the rate of the activity is decreased to get the closest possible to the satefy margin, w.r.t. the minimum rate (Step 12). This avoids oversubscribing the resources while keeping the safety margin. The flow from i to j and the rate of j on the resource are then updated and, since a success is obtained for resource k, the scan of $\mathcal{Q}$ stops (Step 13). If the condition of Step 11 is not reached, the rate of the activity remains too low to keep the safety margin, more flow is needed to decrease its duration. The maximum flow is then set from i to j (Step 15) and the next activity of $\mathcal{Q}$ is considered.

In case that no insertion for j was found after scanning $\mathcal{Q} \cap \mathcal{J}_k$ without violating the safety margin on some resource, a forbidden list is returned. It contains a pair (i, j) with $i \in \mathcal{Q} \cap \mathcal{J}_k$

(Step 17). In case of success, it may happen that different rates and start times were obtained on the resources required by j. In that case, the rate of the activity is set to the maximal one obtained, which also corresponds to the maximal start time due to the continuous convex equality constraint $C_j = S_j + P_j/s_j$ (Step 18). Then, a second step (loop 19–27) aims at increasing the flow sent to activity j on the other resources to have the same activity rate on each resource as required by the problem. This is achieved by scanning again the activities of $\mathcal{Q} \cap \mathcal{J}_k$ for each resource on which rate $s_j^k < s_j$. The activities that complete before S_j are candidate for sending additional flow until the rate is reached. If there is no such activity then a failure is reported by returning a pair (i, j) in the forbidden list $\bar{\mathcal{L}}$ (Step 27).

In case the rate adjustment has been a success, a clustering phase aims at reducing the number of activities sending flow to j in order to diminish the effect of the start time-flow coupling constraints. This is achieved via loop 28–32. If an activity $i \in \mathcal{Q}$ sends flow to j on only one resource k, then the algorithm tries to send the flow units sent by i to j from another activity $i' \in \mathcal{Q}$ already sending flow to j on k.

Then the activity j is added to the already inserted activity list $\mathcal{Q}$ sorted by increasing completion times and the next activity of $\mathcal{L}$ is scanned.

A.1 Illustrative example

The following example shows how the three steps of the elementary activity insertion process inside the INSERT procedure works. Consider a 3 activity instance with two resources k_1 and k_2, each of capacity $R_{k_1} = R_{k_2} = 5$. Each activity $j \in \{1, 2, 3\}$ is such that $s_j^{\min} = 0$, $s_j^{\max} = 5$ and $r_j = 0$. We have $d_1 = 3$, $d_2 = 4$, $d_3 = 6$, $P_1 = 4$, $P_2 = 3$ and $P_3 = 8$. We aim at satisfying a safety margin equal to $\lambda = 2$. We have $\mathcal{J}_{k_1} = \{1, 2, 3\}$ and $\mathcal{J}_{k_2} = \{2, 3\}$. Suppose the input list is $\mathcal{L} = (1, 2, 3)$ and that the already scheduled activities are 1 and 2 as displayed in Figure 8 (a). Hence we visualize that we are at the step of procedure INSERT where $\mathcal{Q} = \{0, 1, 2\}$ with the partial solution $S_1 = 0$, $C_1 = 1$, $s_1 = 4$, $S_2 = 1$, $C_2 = 2$, $s_2 = 3$. The safety margin is satisfied. The resource flows generated up to this point are such that $\Phi_{0,1}^{k_1} = 4 = s_1$, $\Phi_{1,2}^{k_1} = 3 = s_2$. At Step 4 of Algorithm 2, activity 3 is dequeued from $\mathcal{L}$. The maximal flows that activities from $\mathcal{Q}$ can send to activity 3 are then computed at Steps 5 and 6: on resource k_1, we have $\bar{\Phi}_{0,3}^{k_1} = 1$, $\bar{\Phi}_{1,3}^{k_1} = 1$ and $\bar{\Phi}_{2,3}^{k_1} = 3$, and on resource k_2, we get $\bar{\Phi}_{0,3}^{k_2} = 2$, $\bar{\Phi}_{2,3}^{k_2} = 3$. Then the insertion process starts by enumerating for each resource the activities of $\mathcal{Q}$ in increasing order of the completion times (`for` loop at step 9). On resource k_1, insertion after dummy activity 0 yields a solution displayed in part (b) with $auxrate = 1$ that violates the safety margin (checked at Step 11). So the flow $\Phi_{0,3}^{k_1}$ is set to the maximal value $\bar{\Phi}_{0,3}^{k_1} = 1$ and the current rate of activity 3 on resource k_1 is $s_3^{k_1} = auxrate = 1$ (Step 15). Sending flow from activity 1 allows to increase $auxrate$ to 2, which still violates the safety margin (see solution (c)), so again, Step 15 sets $\Phi_{1,3}^{k_1} = \bar{\Phi}_{1,3}^{k_1} = 1$ and $s_3^{k_1} = auxrate = 2$. Adding the flow from activity 2 would allow a total rate $auxrate = 5$, which would make activity 3 end at $2 + 8/5 < 4$. Hence the safety margin is satisfied and an insertion on k_1 has been found. To meet exactly the safety margin, start time $S_3^{k_1}$ is set to 2, rate $s_3^{k_1}$ is set to 4 and flow $\Phi_{2,3}^{k_1}$ is set to 2 at Steps 12 and 13. This gives the solution displayed for resource k_1 on part (d). The same insertion process is then run for activity 3 on resource k_2. In this case, the flow from 0 allows to meet the deadline and thus we get $S_3^{k_2} = 0$, $s_3^{k_2} = 2$ and $\Phi_{0,3}^{k_2} = 2$. The most constraining case corresponds to the maximal start time on all resources and its corresponding rate, also maximal : so S_3 is set to $S_3^{k_1} = 2$ and s_3 is set to $s_3^{k_1} = 4$ at Step 18.

Note that at this point the rate on resource k_2 is too low. We enter the loop at Step 19 to try and adjust the rates. Traversing again list $\mathcal{Q}$ in increasing completion times, activity 2 appears to have enough available outgoing flow to obtain also rate 4 for activity 3 on resource

Algorithm 2: The INSERT procedure

Input: a CTRTP-TW problem instance, an ordered list of activities $\mathcal{L}$, a safety margin λ

Output: a boolean *success*, a solution (S, C, s), a forbidden pair $\bar{\mathcal{L}}$ in case of failure

$\mathcal{Q} \leftarrow \{0\}$ `// list of scheduled activities sorted in increasing completion times`

$\forall k \in \mathcal{R}, \forall i \in \mathcal{J}_k \cup \{0\}, \forall j \in \mathcal{J}_k, i \neq j,\ \Phi^k_{i,j} \leftarrow 0$;

while $\mathcal{L} \neq \emptyset$ **do**

  get last element j from $\mathcal{L}$ and remove it from $\mathcal{L}$;

  $\forall k \in \mathcal{R},\ \bar{\Phi}^k_{0,j} \leftarrow \min(s^{\max}_j, R_k - \sum_{i \in \mathcal{Q} \cap \mathcal{J}_k} \Phi^k_{0,i})$;

  $\forall i \in \mathcal{Q} \cap \mathcal{J}_k, i \neq 0,\ \bar{\Phi}^k_{i,j} \leftarrow \min(s^{\max}_j, s_i - \sum_{i' \in \mathcal{Q} \cap \mathcal{J}_k} \Phi^k_{i,i'})$;

  for $k \in \mathcal{R}_j$ **do**

    $s^k_j = 0$; *successres* $\leftarrow$`false` ;

    for $i \in \mathcal{Q} \cap \mathcal{J}_k$ **do**

      $auxrate \leftarrow \min(s^{\max}_j, s^k_j + \bar{\Phi}^k_{i,j})$;

      if $auxrate \geq s^{\min}_j$ **and** $\max(r_j, C_i) + P_j/auxrate \leq d_j - \lambda$ **then**

        $S^k_j \leftarrow \max(r_j, C_i)$; $auxrate \leftarrow \max\left(s^{\min}_j, P_j/(d_j - \lambda - S^k_j)\right)$;

        $\Phi^k_{i,j} \leftarrow auxrate - s^k_j$; $s^k_j \leftarrow auxrate$; *successres* $\leftarrow$`true`; break;

      else

        $\Phi^k_{i,j} \leftarrow auxrate - s^k_j$; $s^k_j \leftarrow auxrate$;

    if $\neg$*successres* **then**

      $\bar{\mathcal{L}} \leftarrow (i, j)$ for some $i \in \mathcal{Q} \cap \mathcal{J}_k$; **return** $(\texttt{false}, \emptyset, \emptyset, \emptyset, \bar{\mathcal{L}})$;

  $s_j \leftarrow \max_{k \in \mathcal{R}_j} s^k_j$; $S_j \leftarrow \max_{k \in \mathcal{R}_j} S^k_j$; $C_j \leftarrow S_j + P_j/s_j$;

  for $k \in \mathcal{R}_j$ **do**

    if $s^k_j < s_j$ **then**

      for $i \in \mathcal{Q} \cap \mathcal{J}_k,\ \Phi^k_{i,j} < \bar{\Phi}^k_{i,j}$ **do**

        if $C_i \leq S_j$ **and** $s_j - s^k_j \leq \bar{\Phi}^k_{i,j} - \Phi^k_{i,j}$ **then**

          $\Phi^k_{i,j} \leftarrow \Phi^k_{i,j} + s_j - s^k_j$; $s^k_j \leftarrow s_j$; break;

        else if $C_i \leq S_j$ **then**

          $s^k_j \leftarrow s^k_j + \bar{\Phi}^k_{i,j} - \Phi^k_{i,j}$; $\Phi^k_{i,j} \leftarrow \bar{\Phi}^k_{i,j}$

        else

          $\bar{\mathcal{L}} \leftarrow (i, j)$ for some $i \in \mathcal{Q} \cap \mathcal{J}_k$; **return** $(\texttt{false}, \emptyset, \emptyset, \emptyset, \bar{\mathcal{L}})$;

  for $i \in \mathcal{Q}$ **do**

    if $\exists k \in \mathcal{R}_i, \Phi^k_{i,j} > 0$ **and** $\forall k' \in \mathcal{R}_i \setminus \{k\}, \Phi^{k'}_{i,j} = 0$ **then**

      for $i' \in \mathcal{Q} \setminus \{i\}$ **do**

        if $\Phi^k_{i',j} > 0$ **and** $\bar{\Phi}^k_{i',j} \geq \Phi^k_{i,j} + \Phi^k_{i',j}$ **then**

          $\Phi^k_{i',j} \leftarrow \Phi^k_{i,j} + \Phi^k_{i',j}$; $\Phi^k_{i,j} \leftarrow 0$;

  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{j\}$;

return $(\texttt{true}, S, C, s, \emptyset)$;

k_2 by assigning $\Phi_{2,3}^{k_2} = 2$, which gives the feasible solution displayed at part (e).

Although the solution is feasible, we observe that on resource k_1, activity 3 unnecessarily gets flow from all the possible predecessors in $\mathcal{Q}$, which may compromise further insertions. List $\mathcal{Q}$ is scanned a last time in the loop starting at Step 28 to try and improve the flow usage. Activity 1 satisfies the condition checked at Step 30, i.e. activity 1 sends flow to activity 3 but on a unique resource. The other activities of $\mathcal{Q}$ are search to see if they can replace activity 1 on resource k_1, which is the case of activity 2 and we get $\Phi_{1,3}^{k_1} = 0$, $\Phi_{2,3}^{k_1} = 3$. Finally the solution displayed at part (f) is returned.

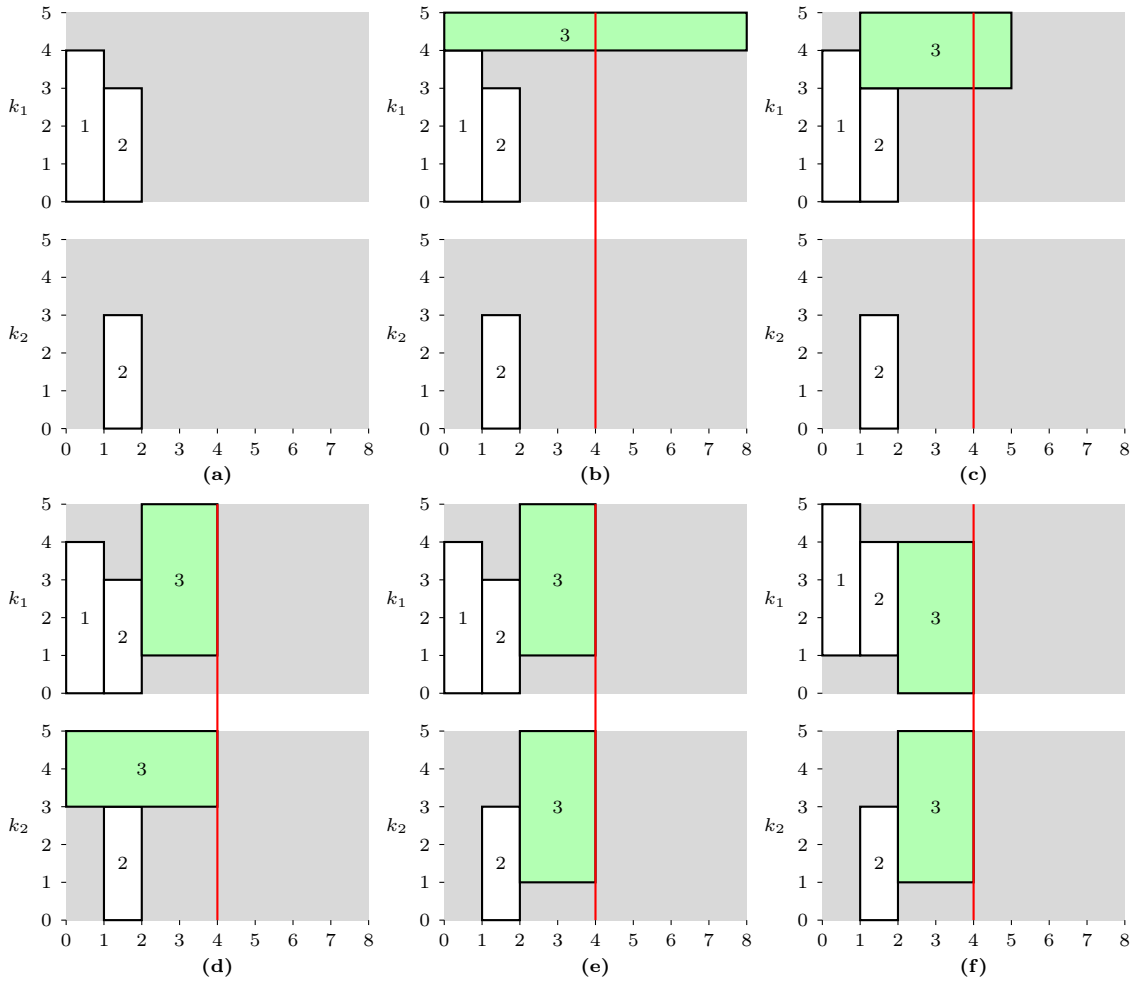


Figure 8: Steps of activity 3 insertion on resources k_1 and k_2

B Branching strategy

At each node, after the upper and lower bounding process based on the binary search described in Section 6.2, the branching process, if needed, is based on the analysis of the preemptive solution

of problem P-CTRTP-TW($\overline{\lambda}, n$), identified by the processing rate s_j^q of each activity $j \in \mathcal{J}$ on each interval $q \in \hat{\mathcal{Q}}$. The branching signature (j^*, α, β), according to the branching scheme described in Section 6.1 is then computed according to the following configurations.

One activity j^* that is processed at a non-constant rate in its processing interval will be selected along with α and β values

Assume that $s_j^0 = s_j^{|\hat{\mathcal{T}}|} = 0$.

We branch in priority on the activities that have not a quasiconcave step rate in the preemptive solution, i.e. the activity rate decreases after some time point q_1 and increases after another time point q_2 ($q_1 < q_2$).

If such activities exist, we select the activity j^* and the ordered pair (q_1, q_2) with the largest value $\Delta = \sum_{q=q_1}^{q_2-1} |s_{j^*}^{q+1} - s_{j^*}^q|$, i.e. the total variation of s_{j^*} (ties are broken lexicographically) and we set $\alpha = t_{q_1+1}$, $\beta = t_{q_2}$. The activity is then included in $\mathcal{M}$.

To illustrate this strategy, Figure 9 show the evolution of the rate of such a "non-quasiconcave" activity. For this activity we have $\Delta = a + b + c$, $\alpha = 16$ and $\beta = 26$.

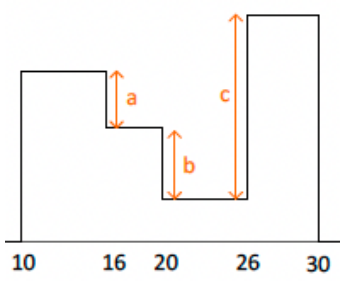


Figure 9: Selection of a "non-quasiconcave" activity for branching

If no such activity exists, we search for a quasiconcave activity that is not in $\mathcal{M}$. Suppose that we find one with increasing or decreasing parts of at least three pieces (type 1 activity).

For such an activity with an increasing part of at least three pieces starting at q_1 and ending at q_2 with $q_1 < q_2$, we denote by $\Delta_j^+ = \sum_{q=q_1}^{q_2-1} s_{j^*}^{q+1} - s_{j^*}^q$ the height of the total rate increase. In the right part of Figure 10, we have $t_{q_1} = 10$ and $t_{q_2} = 20$ and $\Delta_j^+ = a + b$. If the activity has a decreasing part of at least 3 pieces starting at q_1 and ending at q_2, we denote by $\Delta_j^- = \sum_{q=q_1}^{q_2-1} s_{j^*}^q - s_{j^*}^{q+1}$ the height of the total rate decrease. In the left part of Figure 10, we also have $t_{q_1} = 10$ and $t_{q_2} = 20$ and $\Delta_j^- = a + b$.

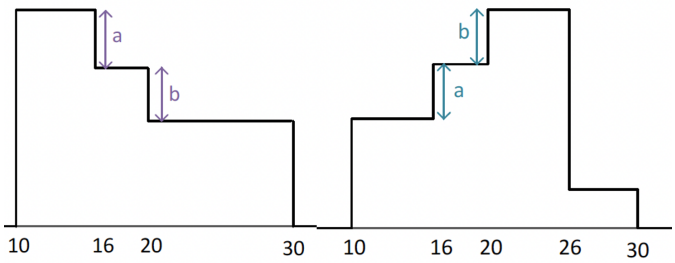


Figure 10: Selection of a quasiconcave activity type 1 for branching

We select an activity with the highest $\Delta_{j^*} = \max(\Delta^+_{j^*}, \Delta^-_{j^*})$ and we set $\alpha = t_{q_1+1}$ and $\beta = t_{q_2}$. In the case of Figure 10, the two activities have the same Δ so one of them is selected with $\alpha = 16$ and $\beta = 20$.

If no such activity exist, we consider the activities for which neither the increasing nor the decreasing part has more than two pieces (type 2 activity, see the two activities displayed in Figure 11). For such activities, we cannot use breakpoints for both α and β.

If the activity has an increasing part of two pieces $(q_1, q_1 + 1)$ (left part of Figure 11, type 2.1 activity) we compute α as the time point such that the same work as in $[t_{q_1}, t_{q_1+1}]$ with rate $s_j^{q_1}$ would be achieved in $[\alpha, t_{q_1+1}]$ with rate $s_j^{q_1+1}$, i.e.

$$s_j^{q_1}(t_{q_1+1} - t_{q_1}) = s_j^{q_1+1}(t_{q_1+1} - \alpha) \Leftrightarrow \alpha = \frac{s_j^{q_1+1} t_{q_1+1} - s_j^{q_1}(t_{q_1+1} - t_{q_1})}{s_j^{q_1+1}}$$

For β, we take t_{q_1+1}. For the activity in the left side of Figure 11, we have $\alpha = \frac{20\times 8 - 6\times(20-10)}{8} = 12.5$ and $\beta = 20$

If the activity has no increasing part of two pieces, it has a decreasing part of two pieces, such as the right activity in Figure 11 (type 2.2 activity). In this case we take symmetrically $\alpha = t_{q_1+1}$ and $\beta = \frac{s_j^{q_1+1}(t_{q_1+2} - t_{q_1+1}) + t_{q_1+1} s_j^{q_1}}{s_j^{q_1}}$. For the right activity of Figure 11, this gives $\alpha = 20$ and $\beta = \frac{8\times(26-20)+20\times 10}{10} = 24.8$.

Among type 2.1 and 2.2 activities, we select randomly one activity in the most populated type.

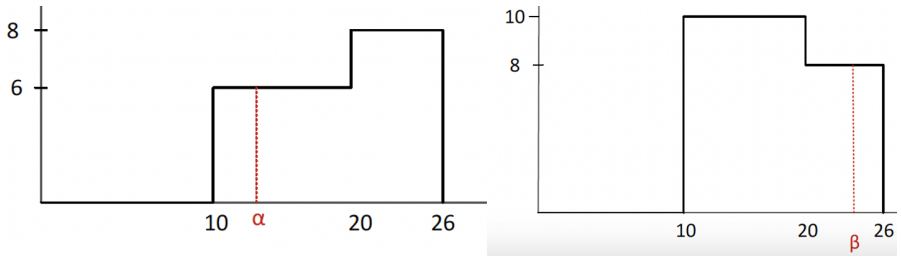


Figure 11: Selection of a quasiconcave activity type 2 for branching

Note that for an activity j belonging to $\mathcal{M}$, we make sure in addition that $[\underline{m}_j, \overline{m}_j] \subset [\alpha, \beta]$

C Precision impact

In this section we evaluate the impact of the precision ϵ on the performance of the dedicated branch-and-bound method presented in Section 6. For three values of ϵ and for the gen1 instances, the column displays the average number of nodes in the tree, the average balue of the lower bound and the average value of the upper bound. Numerical instabilities are observed with a verfy small magnitude for the upper bound but a higher magnitude for the lower bound. This underlines the correctness of the branching scheme while the heuristic used to obtain feasible solutions are more unstable w.r.t. the precision. The number of nodes also fluctuates but the impact on the CPU time is not significant.

	av. #nodes			av. LB			av UB		
ϵ	0.01	0.001	0.0001	0.01	0.001	0.0001	0.01	0.001	0.0001
gen1_15_2	70835.2	65209.5	73015.1	32.712	32.975	32.897	33.366	33.471	33.316
gen1_15_5	53289.1	39781.1	37762.9	37.83	38.527	38.527	40.911	40.945	40.934
gen1_15_10	31875.3	26910.4	23492.6	40.275	40.386	40.275	42.823	42.904	42.925
gen1_20_2	55079	49253.6	49438.4	30.798	31.614	31.706	41.136	41.137	41.137
gen1_20_5	62105.8	39103.3	48431.8	30.557	30.926	30.069	37.403	37.465	37.441
gen1_20_10	55067	40268	44593.1	37.22	37.22	37.22	44.354	44.415	44.416
gen1_25_2	59605.4	47087.6	52534.4	31.869	31.554	31.269	40.299	40.299	40.299
gen1_25_5	32422.4	24241.4	25059.9	30.477	28.905	29.339	38.742	38.742	38.742
gen1_25_10	29299.3	21952	22363.6	28.158	29.391	29.14	50.513	50.604	50.594

Table 8: Impact of the precision on the branch-and-bound performance

D Completeness and finiteness of the branching scheme

In this section, we prove the finiteness and completeness of the branching scheme described in Section 6. We limit ourselves, for the sake of simplicity, to the case when the safety margin λ is fixed so the initial deadline of each activity $j \in \mathcal{J}$ is set to $\hat{d}_j = d_j - \lambda$. Our problem then becomes a satisfiability problem and checking the consistency and finiteness of our search process means checking that:

- If there is no way to schedule the activities of $\mathcal{J}$ while meeting λ, then our algorithm should be able to stop at some time and return a failure;
- If a schedule exists, which meets λ, then our algorithm should be able to stop at some time and return a feasible schedule.

Any node of the search tree is associated with a subdivision of Q intervals $\hat{\mathcal{T}} = \{t_0 = 0, t_1, \ldots, t_Q = T - \lambda\}$, where T is the time horizon. Each t_q is either a release date of an activity, a deadline of an activity, or one of the "middle" points $\underline{m}_j$ and $\overline{m}_j$ of an activity $j \in \mathcal{M}$. Let $Active(j)$ the set of time periods in $\hat{\mathcal{T}}$ for which activity $j \in \mathcal{J}$ is known to be processed at a constant rate, $Idle(j)$ the set of time periods in $\hat{\mathcal{T}}$ for which the activity is known to be absent and $Uncertain(j)$ the activity for which the activity status is undetermined

Lemma 1. *For each activity, the total length of the periods in $Uncertain(j)$ converges to 0 in a finite number of steps for any precision $\epsilon > 0$.*

Proof. We first suppose that at each node, there exists an activity $j \notin \mathcal{M}_i$, for which a branching signature $[\alpha, \beta]$ with $\alpha \geq \hat{r}_j + \epsilon$ and $\beta \leq \hat{d}_j - \epsilon$ is determined according to the procedure described in Appendix B. The branching scheme creates three children nodes: (a) one by increasing the activity realease date to α, (b) one by decreasing the deadline to β and (c) one by inserting this activity in $\mathcal{M}$. A node is obviously eliminated if $r_j > \hat{d}_j - \frac{P_j}{s_j^{\max}} + \epsilon$. Hence, at some finite time, each non eliminated node satisfies one of the two following conditions:

- Due to successive branching (a) and (b), the activity time window is reduced to its minimum feasible valie and $|\hat{d}_j - \frac{P_j}{s_j^{\max}} - \hat{r}_j| \leq \epsilon$. The activity has a fixed rate $s_i^{\max}$ and a fixed execution interval $S_j = \hat{r}_j$ and $C_j = \hat{d}_j$ and the total length of the periods in $Uncertain(j)$ is 0.
- Due to branching (c), the activity belongs to $\mathcal{M}$. The activity rate is a quasiconcave step function, increasing before $\underline{m}_j$, constant in $[\underline{m}_j, \overline{m}_j]$ and decreasing after $\overline{m}_j$.

We can now consider the case of activities that belong to $\mathcal{M}$. For such an activity j, we define as $Idle(j)$ the union of intervals before $[0, \hat{r}_j] \cup [\hat{d}_j, T - \lambda]$, $Uncertain(j)$ the union of intervals $[\hat{r}_j, \underline{m}_j] \cup [\overline{m}_j, \hat{d}_j]$ and $Active(j)$ is the interval $[\underline{m}_j, \overline{m}_j]$. For such an activity j, the branching scheme selects $[\alpha, \beta]$ that strictly includes $[\underline{m}_j, \overline{m}_j]$ with precision ϵ and generates three children nodes, one by updating $\hat{r}_j$ to α, one by updating $\underline{m}_j$ to α and $\overline{m}_j$ to β and one by updating $\hat{d}_j$ to β. In all cases, $Uncertain(j)$ is reduced of at least ϵ. The process being repeated, $Uncertain(j)$ converges to 0 for all $j \in \mathcal{J}$. □

Theorem 4. *The branching process described in Section 6 and Appendix B is consistent and finite.*

Proof. Suppose that there is no feasible schedule and that our algorithm is unable to detect unfeasibility. Consider any path made of nodes Ω_0, Ω_1, Ω_2, ... , Ω_n, ... in the search three where Ω_0 is the root node. The schedule that assigns the exacution window $Active(j)$ and the rate $\hat{s}_j$ computed by the preemptive relaxation for j in $Active(j)$ to each job j is a feasible reduced schedule for $\hat{P}_j = |Active(j)| \times s_j < P_j$. By a compactness argument and Lemma 1, we can define a sequence of nodes where the workload $Active(j) \times \hat{s}_j$ converges to P_j, which yields a feasible schedule, a contradiction. So, for some node n and some $\epsilon > 0$, P-CTRTP-TW(λ, n) will become unfeasible.

Suppose on the opposite that a feasible schedule exists and that the algorithm is unable to find one for any precision $\epsilon > 0$. Let $\sigma = (S, s)$ a feasible schedule. Suppose there is a maximal length path Ω_0, Ω_1, Ω_2, ... , Ω_n, ... in the search three where Ω_0 is the root node such that σ is consistent with Ω_0, ω_1, ... , Ω_n. More precisely, for node Ω_n, we have $Active(j) \subseteq [S_j, C_j]$ and $\hat{r}_j \leq S_j \leq C_j \leq \hat{d}_j$ for each activity j. Let two activities j_1 and j_2 that are overlapping in σ. Then, for some node n and some $\epsilon > 0$, compactness argument and Lemma 1 that j_1 and j_2 will be such that $Active(j_1)$ and $Active(j_1)$ also overlap. Also if j_1 and j_2 do not overlap then intervals $Active(j_1)$ and $Active(j_1)$ are sequenced in the same order. It follows that σ becomes a feasible schedule for CTRTP-TW(π_n), where π_n is the sequencing defined by $Active(j)$ for all activities $j \in \mathcal{J}$. □