Multi-Mode RCPSP with Safety Margin Maximization: Models and Algorithms

Christian Artigues[1], Emmanuel Hebrard[1] Alain Quilliot[2] and Helene Toussaint[2]

[1]*LAAS Laboratory, CNRS, Toulouse, France*

[2]*LIMOS Laboratory, CNRS, Clermont-Ferrand, France*

Keywords: Scheduling, Resource Constrained Project Scheduling Problem, Evacuation Process, Network Flow, Branch and Bound, Linear Programming.

Abstract: We study here a variant of the multimode *Resource Constrained Project Scheduling* problem (RCPSP), which involves continuous modes, and a notion of *Safety Margin* maximization. Our interest was motivated by a work package inside the GEOSAFE H2020 project, devoted to the design of evacuation plans in face of natural disasters, and more specifically wildfire.

1 INTRODUCTION

RCPSP: *Resource Constrained Project Scheduling Problem* (see (Hartmann, 2010), Herroelen, 2005), (Orji, 2013)) involves *jobs* subject to both *temporal* constraints and *cumulative resource* constraints. In *multimode* **RCPSP** (see (Bilseka, 2015), (Weglarz, 2011), resource requirements are flexible and the scheduler may cut a trade-off between speed and resource consumption. The **MSM-RCPSP** (*Multimode with Safety Maximization RCPSP*) model introduced is a variant of *multi-mode* **RCPSP:** for any *job j*, we must choose its *evacuation rate* v_j, which determines, for any resource *e* in the set $\Gamma(j)$ of resources required by *j*, the amount of *e* consumed by *j*. *Release dates* R_j and *deadlines* Δ_j are imposed, and performance is about *safety* maximization, that means the minimal difference (*safety margin*), between job *deadlines* and ending times.

MSM-RCPSP was motivated by the H2020 GEOSAFE European project (GeoSafe, 2018), related to the management of wildfires. At some time during this project, we dealt with evacuation schedules. While in practice evacuation is managed in an empirical way, 2-step optimization approaches have been recently tried (see (Artigues, 2018), and (Bayram, 2016)): the first step (pre-process) identifies the routes that evacuees are going to follow; the second step schedules the evacuation of estimated late evacuees along those routes. This last step implies priority rules and evacuation rates imposed to evacuees and resulting models may be cast into the **MSM-RCPSP** framework.

The paper is structured as follows: Section II describes the **MSM-RCPSP** model. Section III solves the *fixed topology* case. In Section IV we prove that **MSM-RCPSP** *preemptive* relaxation can be solved in polynomial time. We design in Section V and VI both fast heuristic network flow techniques, well-fitted to real-time management, and an exact branch and bound algorithm. Section VII is devoted to numerical tests.

2 MULTI-MODE RCPSP WITH SAFETY MAXIMIZATION

MSM-RCPSP is related to a set *J* of *jobs*, subject to *release dates* R_j and *deadlines* Δ_j, $j \in J$, which have to be scheduled while maximizing what we call the *Safety Margin*. That means that we want to compute starting times T_j and ending times T^*_j in such a way that, for any *job*: $R_j \leq T_j < T^*_j \leq \Delta_j$, and that resulting *Safety Margin*, defined as equal to the quantity $\mathrm{Inf}_{j \in J} (\Delta_j - T^*_j)$, is the largest possible. But we do not know the durations of those *jobs*: as a matter of fact, duration of *j* is determined as a quantity $P(j)/v_j$, where $P(j)$ is some fixed coefficient and v_j is the *speed* of *j*, which is part of the problem

Artigues, C., Hebrard, E., Quilliot, A. and Toussaint, H.
Multi-Mode RCPSP with Safety Margin Maximization: Models and Algorithms.
DOI: 10.5220/0010190101290136
In *Proceedings of the 10th International Conference on Operations Research and Enterprise Systems (ICORES 2021)*, pages 129-136
ISBN: 978-989-758-485-5

and which we also call *evacuation rate* in reference to the *Late Evacuation* problem set in the context of H2020 GEOSAFE project. The choice of those *evacuation rates* is constrained in a cumulative way by the existence of a *resource* set E: any *job* j involves a subset $J(e) \subseteq E$ of resources and at any time between T_j and T^*_j its consumption level of any resource $e \in E$ is equal to the *evacuation rate* v_j, while the amount of available resource e is bounded by a fixed number $CAP(e)$. Let us first link **MSM-RCPSP** with evacuation problems and the GEOSAFE H2020 Program.

2.1 Tree Late Evacuation (Tree-LEP)

We consider here a transit (*evacuation*) network $H = (V, E)$, supposed to be an **oriented tree**:

- *Leaf* subset $J \subseteq V$, called *evacuation node set*, identifies groups of $P(j)$ *j-evacuees* who must reach the *anti-root* safe node *SAFE* while following the arcs of related path $\Gamma(j)$. The last *j-evacuee* must reach *SAFE* before deadline Δ_j. Only one arc $e(j)$ has origin j and only one arc has destination *SAFE*.
- Every arc $e \in E$ is provided with time value $L(e)$, required for any *evacuee* to move along e; *L-length* of $\Gamma(j)$ is denoted by $Length(j)$. Every arc $e \in E$ is also provided with some *capacity* $CAP(e)$: no more than $CAP(e)$ *evacuees* per time unit may enter e at a given time t.

Practitioners impose that all *j-evacuees* move along $\Gamma(j)$ according to the same *evacuation rate* v_j. This *Non Preemption* Hypothesis, makes the *j-evacuation* process to be determined by its starting time T^D_j (when a first *j-evacuee* leaves j), its ending time T^A_j, (when the last *j-evacuee* arrives to *SAFE*) and its *evacuation rate* v_j, subject to (*Evacuation Rate Formula*): $T^A_j = T^D_j + Length(j) + P(j)/v_j$.

Then the ***Late Evacuation Problem*** **(LEP)** consists in the search T^D_j, T^A_j, v_j, $j \in J$, consistent with deadlines and capacities, and maximizing the global *safety margin* $\text{Inf}_j\,(\Delta_j - T^A_j)$.

Example 1: For any arc e in Fig. 1, the first number means the length $L(e)$ and the second one its capacity $CAP(e)$. In case $\Delta_3 = 7$; $\Delta_2 = \Delta_1 = 13$, we make (optimal schedule) group 3 start at time zero according to full rate $v_3 = 2$, and both groups 1 and 2 start at time 4, according to rates $v_1 = v_2 = ½$.

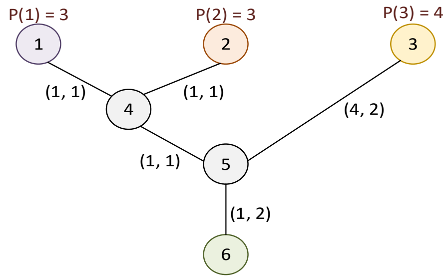

Figure 1: An Instance of Tree-LEP.

Figure 2 represents related optimal schedule according to a Gantt diagram: The height of rectangle j is the *evacuation rate*; its width is delimited by the time when population j starts entering node 6 and the time when it has finished.

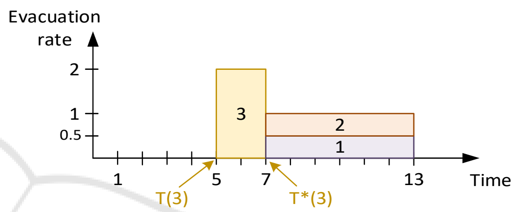

Figure 2: TREE-LEP Schedule in RCPSP format.

In order to turn a **Tree-LEP** solution into **RCPSP** format, we set, for any j in J: $R_j = Length(j)$ and $vmin_j = P(j)/(\Delta(j) - R_j))$. *Seadline* constraint implies $v_j \geq vmin_j$; $vmax_j = CAP(e(j))$. Then we consider the process defined by the *j-evacuees* when they enter into the *SAFE* node, and call it *evacuation job j*. Its starting time is $T_j = T^D_j + Length(j)$, its ending time is $T^*_j = T^A_j$ and we want to **maximize** *Safe-Margin* = $\text{Min}_{\,j \in J}\,(\Delta_j - T^*_j)$. If v_j denotes related *evacuation rate*, we get the following **temporal** constraints: $R_j \leq T_j \leq T^*_j \leq \Delta_j$ and $T^*_j = T_j + P(j)/v_j$. As for resource constraints, we say that 2 jobs j_1, j_2, *overlap* iff interval $[T_{j1}, T^*_{j1}] \cap [T_{j2}, T^*_{j2}]$ is neither empty nor reduced to one point. Then **resource** constraints tells that for any arc e in A and for any *Overlap* clique $J_0 \subseteq J(e) = \{j$ such that $e \in \Gamma(j)\}$, we should have: $\Sigma_{\,j \in J0 \cap J(e)}\, v_j \leq CAP(e)$. In case $J_0 = e(j)$, this yields $v_j \leq vmax_j$.

2.2 The MSM-RCPSP Model

According to 2.1, **MSM-RCPSP Inputs** are: The *job* set J and the resource set E; for any $j \in J$, *Population* coefficient $P(j)$, *Release* date R_j, *Deadline* Δ_j, maximal *evacuation rate* $vmax_j$ and set subset $\Gamma(j) \subseteq E$ of resources used by j; for any $e \in E$, the *Capacity* $CAP(e)$ = and the subset $J(e) \subseteq E$ of *jobs* j which use e. Then **MSM-RCPSP** model, conjectured to be NP-Hard, comes as follows:

MSM-RCPSP Model: Compute Rational Vectors $T = (T_j, j = 1..N)$, $T^* = (T^*_j, j = 1..N)$, $v = (v_j, j = 1..N) \geq 0$, and $\{0, 1, -1\}$**-valued vector** $\Pi = (\Pi_{j1,j2}, j_1, j_2 = 1..N)$ with **Semantics** : $\Pi_{j1,j2} = 1 \sim j_1 << j_2$; $\Pi_{j1,j2} = -1 \sim j_2 << j_1$; $\Pi_{j1,j2} = 0 \sim j_1$ *Overlap* j_2, such that:

- **Structural Constraints:** For any j_1, j_2, $\Pi_{j1,j2} = - \Pi_{j2,j1}$.
- **Temporal Constraints:**
  - For any j: $R_j \leq T_j \leq T^*_j \leq \Delta_j$ and $T^*_j = T_j + P(j)/v_j$; (E1)
  - For any pair j_1, j_2, the following implication holds: $\Pi_{j1,j2} = 1 \rightarrow T_{j2} \geq T^*_j$; (E1*)
- **Resource Constraints:**
  - For any j : $vmin_j = P(j)/(\Delta(j) - R_j)) \leq v_j \leq vmax_j$;
  - For any arc e, (E2) implication holds: (E2) ($J_0 \subseteq J$ is such that for any pair j_1, j_2 in J_0, $\Pi_{j1,j2} = 0$) -> $\Sigma_{j \in J0 \cap J(e)} v_j \leq CAP(e)$;
- **Maximize** : *Safe-Margin* = $\mathrm{Inf}_j (\Delta(j) - T^*_j)\}$

This model fits with industrial contexts, where jobs j involving continuous flows of items are applied a sequence $\Gamma(j) = \{e^j_1, e^j_2, .., e^j_{n(j)}\}$ of operations, and pipe-lined through some set of machines.

3 FIXING THE TOPOLOGY

It will happen in next sections that we are provided with some *topological* vector Π. So we denote by **MSM-RCPSP**(Π) resulting **MSM-RCPSP** model. **MSM-RCPSP**(Π) model is convex. In order to linearize **MSM-RCPSP**(Π), we replace, for any j, T^*_j by $T_j + P(j)/v_j$, and reformulate (E1) as:

- For any j, and any $w \in [vmin_j, vmax_j]$: $(\Delta_j - RMin - T_j) \geq (- v_j + w).P(j)/w^2 + P(j)/w$.

This constraints tells us that for any w the 2D-point $(v_j, (\Delta_j - T_j - RMin))$ must be located above the tangent line in $(w, P(j)/w)$ to the hyperbolic curve whose equation is $x \rightarrow P(j)/x$. We proceed the same way with E1* and get the following linear formulation **LINEAR-MSM-RCPSP**(Π):

LINEAR-MSM-RCPSP(Π): {**Compute** $T = (T_j, j = 1..N)$, $v = (v_j, j = 1..N) \geq 0$ and $RMin \geq 0$, s.t:

- **Temporal constraints:**
  - For any j : $R_j \leq T_j$;
  - For any j and any $w \in [vmin_j, vmax_j]$: (E1) $(\Delta_j - Rmin - T_j) \geq (- v_j + w).P(j)/w^2 + P(j)/w$;
  - For any j_1, j_2 s.t $\Pi_{j1,j2} = 1$, any $w \in [vmin_{j1}, vmax_{j1}]$: (E1*) $T_{j2} - T_{j1} \geq (- v_{j1} + w).P(j_1)/w^2 + P(j_1)/w$;
- **Capacity Constraints:**
  - For any j : $vmin_j \leq v_j \leq vmax_j$;
  - For any e, any subset $J_0 \subseteq J$ s.t for any j_1, j_2 in J_0, $\Pi_{j1,j2} = 0$: $\Sigma_{j \in J0 \cap J(e)} v_j \leq CAP(e)$; (E2)
- **Maximize:** *Safe-Margin* = $RMin$}.

We apply a *cutting plane* process to (E1, E1*):

Linear-MSM-RCPSP-Cut(Π):
Initialize a set W of constraints (E1, E1*) and condider related restriction **LINEAR-MSM-RCPSP**(Π, W); Not *Stop* ;
While Not *Stop* **do**
  Solve **LINEAR-MSM-RCPSP**(Π, W);
  Search for j_0 (j_1, j_2) and w_0 such that (E1, E1*) do not hold;
  If *Fail*(Search) **then** *Stop*
  Else Insert (E1, E1*) related to j_0, w_0 into W.

4 PREEMPTIVE MSM-RCPSP

Preemptive **MSM-RCPSP** means that *jobs* may stop at some time and start again a little later. *Preemption* allows any *job* j to be split into $k(j)$ sub-processes $j_1,.., j_{k(j)}$, each with starting time $t_{j,k}$, ending time $t^*_{j,k}$, and *evacuation rate* $v_{j,k}$. We denote by **P-MSM-RCPSP** the resulting problem. Figure 3 below shows an example of *preemptive* schedule related to example 1.

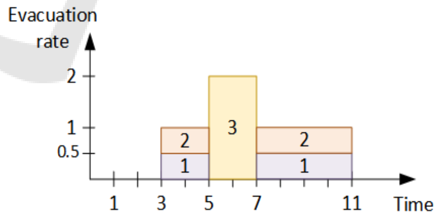

Figure 3: P-MSM-RCPSP Schedule.

Let us now suppose that we are provided with some *safety margin* $\lambda \geq 0$ which we want to ensure. Then we set $S = \{R_j, (\Delta_j - \lambda), j \in J\}$ and label its elements $\{t_1,.., t_{2N}\}$, in such a way that $t_1 \leq t_2 \leq \ldots \leq t_{2N}$. For any $k = 1,\ldots,2N\text{-}1$, we set $\delta_k = t_{k+1} - t_k$. This leads to the following rational PL **Preemptive**(λ):

Preemptive(λ) **Linear Program: Compute** rational vector $w = (w_{j,k}, j \in J, k = 1..2N\text{-}1) \geq 0$, whose semantics is that $w_{j,k}$ is the evacuation rate for j between t_k and t_{k+1}, and which satisfies the

- For any j, k, $w_{j,k} \leq vmax_j$;
- For any j, $\Sigma_k \, \delta_k . w_{j,k} = P(j)$;
- For any arc e, any k: $\Sigma_{j \in J(e)} w_{j,k} \leq CAP(e)$;
- For any j and any k such that $t_{k+1} \leq R_j$, $w_{j,k} = 0$;
- For any j, k such that $t_k \geq (\Delta_j - \lambda)$: $w_{j,k} = 0$}.

Lemma 1: ***Preemptive(λ)*** *identifies a preemptive schedule which is consistent with safety margin λ, in case such a schedule exists.*

Proof: If a *preemptive* schedule exists, consistent with *safety margin* λ, release dates R_j, deadlines Δ_j, $j \in J$, and capacities $CAP(e)$, $e \in A$, then it can be chosen in such a way that for any *job j* and any k, related *evacuation rate* of j is constant between t_k and t_{k+1}. Then we get above linear program. □

We solve **P-MSM-RCPSP** by applying the following binary process ***Optimal-P-MSM-RCPSP***, which computes optimal *safety margin λ-Val* by making λ iteratively evolve between a non feasible value λ_1 and a feasible one λ_0:

Optimal-P-MSM-RCPSP(*Threshold*):
λ_0 <- 0 ; λ_1 <- Inf$_j$ [$\Delta(j) - (R_j + P(j)/vmax_j)$]; *w-Sol* <- *Nil* ; *λ–Val* <- - ∞ ; Solve **Preemptive**(λ_1);
If *Success*(Solve) **then** *λ–Val* <- λ_1; *w-Sol* <- related vector *w*
Else
  Solve **Preemptive**(λ_0);
  If *Success*(Solve) **then**
    λ–Val <- λ_0; *w-Sol* <- related vector *w* ; *Counter* <- 0;
    While *Counter* ≤ *Threshold* **do**
      λ <- $(\lambda_1+\lambda_0)/2$; Solve **Preemptive**(λ);
      If *Success*(Solve) **then** λ_0 <- λ ; *λ–Val* <- λ_0; *w-Sol* <- related *w* Else λ_1 <- λ ;
      Optimal-P-MSM-RCPSP <- (*λ–Val*, *w-Sol*);
  Else *Optimal-P-MSM-RCPSP* <- *Fail*;

Theorem 1: ***Optimal-P-MSM-RCPSP*** *solves the* ***P-MSM-RCPSP*** *Problem in Polynomial Time.*

Proof: Optimality comes in straightforward way from the very meaning of linear program **Preemptive**(λ). As for complexity, we set *Threshold* = Log_2(Sup $_j$ Maximal binary encoding size of Δ_j and R_j + 1) and derive Time-Polynomiality from time polynomiality of LP.€

Sterilization: We may try to turn *w* into a *non preemptive* schedule through 2 approaches:

- ***Sterilization1***: *Smoothing w* while keeping *safety margin* λ as in Figure 4 below:

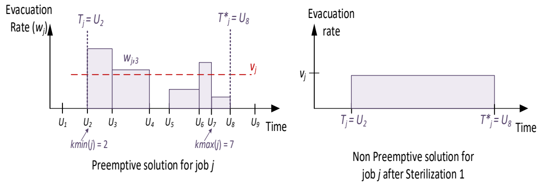

Figure 4: Sterilization1 Scheme.

- ***Sterilization2***: Deriving from *w* a topological vector Π, and solving **MSM-RCPSP**(Π).

5 A FLOW BASED HEURISTIC

This section is devoted to the description of a network flow based heuristic, which implements insertion mechanisms as in (Quilliot, 2012), and computes an efficient feasible **MSM-RCPSP** solution. We consider resources *e* as flow units, that *jobs j* share or transmit: If we represent every *job* as a rectangle whose length is the duration $T^*_j - T_j$ and height is the *evacuation rate* v_j, then, if j_1 precedes j_2, and if not *jobs j* is located between j_1 and j_2 on the *e*-diagram, then we see (fig. 2 and 6) that part of *evacuation rate* v_{j1} related to resource *e* is transmitted to j_2. In order to formalize this, we build an auxiliary network *G* in which the vertex set is $J \cup \{s, p\}$, where *s* and *p* are two fictitious jobs *source* and *sink*, whose arcs are all arcs (i, j), $i, j \in J$, augmented with all arcs (s, j) and all arcs (j, p). Then we consider that the backbone of a schedule is a flow vector $w = (w^e_{j1,j2}, j_1, j_2 \in J(e) \cup \{s, p\}) \geq 0$, which represents, for all resources *e*, the way *jobs* share resource *e*. Clearly, this vector *w* must satisfy standard **flow conservation** laws:

- For any *e*: $\Sigma_{j \in J} w^e_{s,j} = \Sigma_{j \in J} w^e_{j,p} = w^e_{p,s} = CAP(e)$;
- For any resource *e* of *E* and any *job* $j_0 \in J(e)$, $\Sigma_{j \in J \cup \{p\}} w^e_{j0,j} = \Sigma_{j \in J \cup \{s\}} w^e_{j,j0} = v_{j0}$.

Besides, if we introduce starting times T_j and ending times T^*_j as in II, then, for any j_1, j_2, the following implication is true: $\Sigma_e w^e_{j1,j2} \neq 0$ -> $T_{j2} \geq T^*_{j1}$. This ***logical*** constraint means that if *job* j_1 provides j_2 with some part of resource *e*, then j_1 should be achieved before j_2 starts. Clearly, we must keep on with the other ***standard constraints***:

- For any *j*: $R_j \leq T_j \leq T^*_j \leq \Delta_j$; $v_j \leq vmax_j$; $T^*_j = T_j + P(j)/v_j$; $T_s = T^*_s = 0$.
- **Maximize** Min$_j$ $(\Delta_j - T^*_j)$.

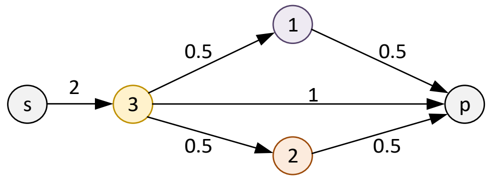

Figure 5: Example 2, with $\Delta_3 = 7$, $\Delta_1 = \Delta_2 = 13$.

5.1 An Adaptative Insertion Heuristic

We deal with **MSM-RCPSP-Flow** through an insertion algorithm which manages two antagonistic trends: when handling *job j* and trying to insert it into a current partial schedule (T, T^*, v), we first compute T_j and next assign v_j a value. But if we choose a high value v_j in order to make j finish fast, then we may block the access to the most critical resources of $\Gamma(j)$. In order to find a compromise we control an *adaptative safety margin* λ through binary search and a related *adaptative priority list* σ, which drives the insertion process for a given λ. For a given value of λ, and a current list σ, the procedure ***Insert-MSM-RCPSP*** (λ, σ) scans the *jobs* j_0 in σ, and try to compute T_{j0} and v_{j0} in such a way that **MSM-RCPSP-Flow** constraints are satisfied for all *jobs j* before or equal to j_0 according to σ, and that v_{j0} is minimal. In case of success, then λ is increased, else ***Insert- MSM-RCPSP***(λ, σ) yields a set of pairs j_1, j_2, asked to become such that $j_2 \ \sigma \ j_1$ (Instruction ***Update***(σ) below).

MSM-RCPSP-Flow(*Precision*:Number)
Algorithmic Scheme:
Step 1: Start from a non feasible margin λ-*max*, a feasible one λ-*min*, a related *Current-Schedule*; *Initialize priority list* σ: priority given to *jobs j* with small $P(j)$ and expected *safety*;
While (λ-*max* - λ-*min*) $\geq$ *Precision* **do**
  λ <- (λ-*min* + λ-*max*)/2; ***Insert-MSM*** (λ, σ)
  If *Success* **then** set λ-*min* to λ and *Update Current-Schedule*
  Else ***Update***(σ); *Retrieve* topology Π from *Current-Schedule*;
  Step2: *Solve* resulting **P-MSM-RCPSP**(Π).

5.2 Insert-MSM Procedure

This procedure works while scanning current priority list σ and assigning T_j and v_j values as far as *jobs j* come. That means that at any time during the process, we are considering some *job* j_0, while all *jobs j* such that $j \ \sigma \ j_0$ have been scheduled: for any $j \in J \cup \{s\}$ such that $j \ \sigma \ j_0$, we are provided with values T_j, T^*_j, v_j, as well as with values $\Phi(e, j)$ which represents the amount (*evacuation rate*) of *e*-resource that j is able to transmit to j_0, according to flow vector w^e of the **MSM-RCPSP-Flow** model. Then we proceed in 3 steps:

- **<u>1st Step</u>:** Scan $\Gamma(j_0)$ according to decreasing $\Phi(e, j_0)$ values, and for any e in $\Gamma(j_0)$, provide j_0 with an amount of resource e in such a way resulting T^*_{j0} does not exceed Δ_{j0} - λ.
- **<u>2nd Step</u>:** In case of success of previous first step, we become provided with an *evacuation rate* v_{j0} and, for any resource $e \neq e_0$ in $\Gamma(i_0)$ with an *evacuation rate* value *v-aux*$_e$ which may be less than v_{i0}; So the second step makes increase the values $w^e_{j,i0}$ for any $e \neq e_0$, $j \in J(e)$, in order to make j_0 run according to the same *evacuation rate* for all arcs e of $\Gamma(j_0)$.
- **<u>3rd Step</u>:** In case of success of previous second step, last step is a *clustering* step, which aims at making decrease the number of resources provided with non null $w^e_{j,i0}$ values, and works by shifting, as far as possible, values $w^e_{j,i0}$ which involve, for a given j, only one resource e, to another *job j'* such that $j' \in J(e)$, $w^e_{j',i0} \neq 0$ and $\Pi(e, j') \geq w^e_{j,i0} + w^e_{j',i0}$.

Example 2: Suppose that we face here the following situation: $\sigma = s, \ldots, j_1, \ldots, j_2, \ldots j_3, \ldots, j_0$; $\Gamma(x_0) = \{e_1, e_2\}$; $CAP(e_1) = 20$, $CAP(e_2) = 25$; $\Delta_{j0} = 21$; $P(j_0) = 5$; $R_{j0} = 10$; $j_1 \in J(e_1) \cap J(e_2)$; $j_1 \in J(e_1)$; $j_3 \in J(e_2)$; $P(j_1)/v_1 = 6$; $P(j_2)/v_2 = 3$; $P(j_3)/v_3 = 4$.

⇨ Then we get:
Step1 -> $w^{e1}_{s,j0,} = 2$; $w^{e2}_{s,j0} = 3$; $w^{e1}_{j1,j0} = 8$; $v_{j0} = 10$; *Success*;
Step2 -> $w^{e2}_{j2,j0} = 7$; *Success*; *Step3 ->* $w^{e1}_{j1,j0} = 0$; $w^{e1}_{j2,j0} = 8$; $T_{j0} = 21$.

6 AN EXACT ALGORITHM

This *Branch&Bound* algorithm relies on sections IV and V: *Optimistic* estimation (upper bound) derives from IV, and an initial feasible solution is computed according to V. We must specify:

- The nodes of related *search tree* and the way *optimistic estimation* is adapted to those nodes;
- The *Branching Strategy* and the global *Tree Search* process.

***The nodes of the Search Tree*:** Such **a node** *s* will be defined by a *Release* vector $A = (A_j, j \in J) \geq R = (R_j = j \in J)$, a *Deadline* vector $B = (B_j, j \in J) \leq \Delta$

and 2 partially defined *Medium* vectors $U = (U_j, j \in J(s))$, $U^* = (U^*_j, j \in J(s)$ such that :

- $J(s)$ denotes the set of *jobs* j such that U_j and U^*_j are defined;
- If $j \in J(s)$, then $A_j \leq U_j < U^*_j \leq B_j$;

For a given *job* j, the meaning of U_j and $U^*_j, j \in J(s)$ is that $w_j = w_j(t)$ must be constant on $[U_j, U^*_j]$ and such that, for any t' outside $[U_j, U^*_j]$, t inside $[U_j, U^*_j]$, $w_j(t) \geq v_j(t')$. Then **Branching from** ***s***. Given a *job* j and 2 values α and β such that $A_j < \alpha < \beta$, node s gives rise to 3 *sons*:

- **First *son*:** A_j is replaced by α;
- **Second *son*:** B_j is replaced by β;
- **Third *son*:** U_j is replaced by α and U^*_j by β: we must have: $\alpha < U_j < U^*_j < \beta$.

The 3-uple (j, α, β) defines the *Branching Signature*. Once created, node s is applied an *optimistic estimation* procedure, and next, in case *Sterilization* does not work, stored into a *Breadth-First Search* list together with resulting value λ-*Val* and related *Branching Signature Sign* = (j_0, α_0, β_0).

Optimistic Estimation and Sterilization Procedures: They derive from Section IV: we solve **P-MSM-RCPSP** augmented with additional constraints related to node s. More precisely:

- For any j, we set $B^*_j = \text{Inf}\,(B_j, \Delta_j - \lambda)$ and $S = \{A_j, B^*_j, j \in J\} \cup \{U_j, U^*_j, j \in J(s)\}$. We order $S = \{t_1,\ldots, t_k, \ldots t_K\}$ through increasing values $t_1 < t_2 < \ldots < t_K$ and set, for any $k = 1..\ K\text{-}1$: $\delta_k = t_{k+1} - t_k$
- We build 4 vectors *k1, k2, k3, k4*, with indexation on J, and whose meaning is:
  - $k1_j$ means the value k such that $A_j = t_k$;
  - $k2_j$ means the value k such that $B^*_j = t_k$;
  - $k3_j$ means the value k such that $U_j = t_k$; (* If U_j is undefined, then $k3_j = 0$*)
  - $k4_j$ means the value k such that $U^*_j = t_k$; (*If U_j is undefined, then $k4_j = 0$*)

According to this, we adapt the program **Preemptive**(λ) to node s by setting:

Preemptive^s(λ): {Compute $w = w_{j,k}$, $j \in J$, $k = 1..K\text{-}1\}$ such that;

- For any e and any k, $\Sigma_{j \in J(e)}\, w_{j,k} \leq CAP(e)$
- For any j, $\Sigma_k\, \delta_k \,.\, w_{j,k} = P(j)$
- For any j, any $k \leq k1(j) - 1$, $w_{,k} = 0$;
- For any j, any $k \geq k2(j)$, $w_{j,k} = 0$;
- For any j, any $k \geq k3(j)$, $w_{,k+1} \leq w_{,k}$;
- For any j, any $k \leq k4(j)\text{-}2$, $w_{j,k+1} \geq w_{j,k}$.

We try to turn a solution of **Preemptive**^{s}(λ) into a **MSM-RCPSP** Solution through procedures ***Sterilizationx*** , $x = 1, 2$ of IV, and adapt ***Optimal-P-MSM-RCPSP*** into a procedure ***UB*** in order to make it compute, for a given node $s = (A, B, U, U^*)$, related *optimistic estimation* λ-*Val* = ***UB***(s)**.**

Branching Strategy**:** Let us suppose that we just computed λ-*Val* = ***UB***(s)**,** got a preemptive solution w, which we could not turn into a *non preemptive* solution with better *Safety Margin* than our current best feasible value. Then, for any job j, we scan the index set $1..K$, and compute a word $\Sigma^j = \{\Sigma^j_1,.., \Sigma^j_K\}$ representative of the *resource profile* induced by j:

- If $\varepsilon = 1$ then $w_{j,k} > w_{j,k-1}$;
- If $\varepsilon = -1$ then $w_{j,k} < w_{j,k-1}$;
- If $\varepsilon = 0$ then $w_{j,k-1} = w_{j,k}$ and $h = 0$.

This word Σ^j enables us to identify:

- 1st **Configuration:** A *hole* (see Fig. 6) with some *depth* and *width* and a *weight* = *depth.width;*
- 2sd **Configuration:** No *hole* but a *left stair* or a *right stair* with once again a *depth*, a *width* and a *weight*.

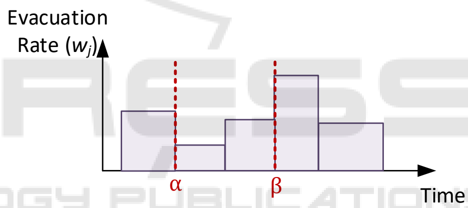

Figure 6: Hole (1st Configuration) Branching.

So our **Branching Strategy** comes as follows: In case Configuration 1, then we compute *branching signature Sign* as some related *Sg* with largest *weight* = *depth.width*. In case it does not exist, then we look for *Sg* related to configuration 2 with the highest *weight* value.

Resulting Branch and Bound Algorithm B&B-MSM-RCPSP: B&B- MSM-RCPSP is implemented as follows, according to a BFS (*Breadth First Search*) strategy. In case of interruption, we get a lower bound *BInf* and an upper bound *BSup*.

7 NUMERICAL EXPERIMENTS

Technical Context: Algorithms are implemented in C++, gcc 7.3. Linear models are solved with Cplex 12.8. Hardware involves Processors Intel(R)

Xeon(R) CPU E7-8890 v3 @ 2.50 GHz, run by Linux.

Instance Generation: Instances come from the GEOSAFE project (see (Artigues, 2018)). They are clustered into 10 instance groups *dense_x*, *medium_x*, *sparse_x*, where *x* is the number of *jobs*, and *dense, medium* and *sparse* are related to the mean degree of the nodes in related tree.

Table 1: Characteristics of the Instances.

Instances	*Nodes*	*Cap-Relax*	*Congest*
dense_10	19.80	155.06	1.69
dense_15	29.10	160.08	1.78
dense_20	38.60	164.88	1.84
medium_10	19.70	152.83	1.71
medium_15	29.10	159.39	1.80
medium_20	38.20	160.69	1.86
medium_25	46.80	169.91	1.91
sparse_10	19.50	146.17	1.75
sparse_15	28.80	153.92	1.87
sparse_20	38.30	157.78	1.87
sparse_25	47.60	154.73	1.89

For any 10 instances group, above Table 1 provides us with: the mean number *Nodes* of nodes, the minimal duration *Cap-Relax* of the evacuation process in case capacity constraints are relaxed and the mean (for all nodes *x*) ratio *Congest*, between the sum of capacities of the in-arcs and the capacity of the out-arc related to *x*.

7.1 Evaluating Optimal-P-MSM-RCPSP and MSM-RCPSP-Flow

We focus here on the ability of ***Optimal-P- MSM-RCPSP*** and ***MSM-RCPSP-Flow*** to provide us with a good **MSM-RCPSP** Lower/Upper approximation window. Table 2 provides, for every instance group:

- ***Opt-P-MSM*****:** Optimal *safety margin* (***Optimal-P-MSM-RCPSP***); ***Opt-P*****-CPU**: Related CPU time;
- ***# fails*****:** the number of instances for which ***MSM-RCPSP-Flow*** yields a *fail* result;
- ***MSM-Flow*****:** *Safety margin* computed by ***MSM-RCPSP-Flow***; ***Flow-CPU***: Related CPU Time;
- ***Preempt-Gap*****:** the gap between ***MSM-Flow*** and the ***Opt-P-MSM***.

Table 2: Behavior of MSM-RCPSP-Flow.

Instances	*Opt-P-MSM*	*Opt-P-CPU*
dense_10	106.25	0.05
dense_15	68.3	0.08
dense_20	39.29	0.11
medium_10	92.59	0.04
medium_15	65.35	0.08
medium_20	54.85	0.11
medium_25	49.55	0.32
sparse_10	113.78	0.04
sparse_15	78.33	0.06
sparse_20	64.45	0.11
sparse_25	21.69	0.45

Table 3: Bis: Behavior of MSM-RCPSP-Flow.

Instances	*MSM-Flow*	*Preempt-Gap*	*Flow-CPU*	*# fails*
dense_10	97.09	9.23	2.01	0
dense_15	58.02	20.96	2.31	0
dense_20	34.75	20.03	2.97	2
medium_10	88.76	4.21	2.06	0
medium_15	52.87	18.24	2.82	0
medium_20	43.75	20.85	3.53	2
medium_25	36.78	26.87	1.96	1
sparse_10	110.38	3.65	1.88	0
sparse_15	75.77	3.24	2.75	0
sparse_20	48.70	28.50	3.94	0
sparse_25	32.67	*	1.18	4

Comment: ***Optimal-P-MSM-RCPSP*** and ***MSM-RCPSP-Flow*** provide us with respectively efficient optimistic and realistic approximations.

7.2 Evaluating B&B-MSM-RCPSP

We focus here on the filtering process and the number of nodes of the *search tree* which are visited during the process. We compute (Table 3):

- The value ***Opt-P-MSM*** as in Table 2;
- The lower (feasible) bound ***B&B- MSM-Inf*** provided by ***B&B-MSM-RCPSP***; The lower bound ***B&B-MSM-Sup*** provided by ***B&B-MSM-RCPSP***; Related CPU time ***B&B- CPU*****;**
- The number ***Nodes o***f nodes of the *search tree* which were visited during the process.

Table 4: Behavior of B&B- MSM-RCPSP.

Instances	*Opt-P-MSM*	*B&B-MSM- Inf*	*B&B-MSM-Sup*
dense_10	106.25	105.75	105.76
dense_15	68.30	66.65	67.49
dense_20	39.29	38.86	39.29
medium_10	92.58	91.87	91.87
medium_15	65.35	63.42	64.48
medium_20	54.85	49.82	54.57
medium_25	51.60	49.19	51.60
sparse_10	113.78	113.75	113.78
sparse_15	78.33	78.33	78.33
sparse_20	64.45	59.21	64.38
sparse_25	34.34	32.49	34.34

Table 5: Bis: Behavior of B&B-MSM-RCPSP.

Instances	*Nodes*	*B&B-CPU*
dense_10	12077.80	361.14
dense_15	27148.80	1090.40
dense_20	11338.90	720.85
medium_10	4253.30	105.51
medium_15	28850.20	1440.62
medium_20	27254.90	1800.03
medium_25	10839.00	1081.33
sparse_10	57425.50	604.36
sparse_15	10668.30	360.32
sparse_20	15164.50	1440.08
sparse_25	17054.10	1800.28

Comment: ***B&B-MSM-Inf*** is always very close to optimistic estimation ***Opt-P-MSM***, and ***Optimal-P-MSM-RCPSP*** provides us with a very good approximation of optimality. Still, it is difficult to make this optimistic estimation decrease.

8 CONCLUSIONS

We introduced here a **Multi-Mode RCPSP** model with both discrete and continuous features, solved its *preemptiv*e version, proposed a network flow based heuristic as well as an exact Branch&Bound algorithm. Further work will aim at extending the model and exploring potential industrial applications.

ACKNOWLEDGEMENTS

We thanks E.U Community for funding H2020 GeoSafe Project.

REFERENCES

Artigues.C, Hebrard.E, Pencolé.Y, Schutt.A, Stuckey.P, 2018. A study of evacuation planning for wildfires; *ModRef 2018 Workshop,* Lille, France.

V.Bayram.V, 2016. Optimization models for large scale network evacuation planning and management; *Surveys in O.R and Management Sciences*.

Bilseka.U, Bilgea.Ü, Ulusoy.G, 2015. Multi-mode resource constrained multi-project scheduling and resource portfolio problem; *EJOR*, 240, 1, p 22-33.

Geo-Safe- *geospatial based environment for optimization systems addressing fire emergencies*; *MSCA-RISE 2015 H2020,* http://fseg.gre.ac.uk/fire/geo-safe.html. Accessed Jue 12, (2018).

Hartmann, S., Briskorn, D., 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. *EJOR*, 207: 1-14.

Herroelen, W., 2005. Project scheduling - theory and practice. *Production and Operations Management*, 14(4): 413-432.

Orji.M.J, Wei.S, 2013. Project Scheduling Under Resource Constraints: A Recent Survey. *IJERT* Vol. 2 Issue 2.

Quilliot.A, Toussaint.H, 2012. *Flow Polyedra and RCPSP*, *RAIRO-RO*, 46-04, p 379-409.

Weglarz. J, Jozefowska. J, Mika.M, Waligora.G, 2011. Project scheduling finite/infinite processing modes - A survey. *EJOR*, 208(10): 177-205.