Multi-Mode RCPSP with Safety Margin Maximization: Models and Algorithms
Christian Artigues, Emmanuel Hébrard, Alain Quilliot, Hélène Toussaint

To cite this version:
Christian Artigues, Emmanuel Hébrard, Alain Quilliot, Hélène Toussaint. Multi-Mode RCPSP with Safety Margin Maximization: Models and Algorithms. 10th International Conference on Operations Research and Enterprise Systems, Feb 2021, Online Streaming, Austria. pp.129-136, 10.5220/0010190101290136 . hal-03160056

HAL Id: hal-03160056
https://laas.hal.science/hal-03160056
Submitted on 4 Mar 2021

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

Christian Artigues¹, Emmanuel Hebrard¹ Alain Quilliot² and Helene Toussaint²

¹LAAS Laboratory, CNRS, Toulouse, France
²LIMOS Laboratory, CNRS, Clermont-Ferrand, France
{artigues, hebrard}@laas.fr, {alain.quilliot, helene.toussaint}@isima.fr

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 \( \Delta_j \) 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 \( \Delta_j, j \in J \), 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 \( \inf \sum_{j \in J} (\Delta_j - T^*_j) \), 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 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 \( \text{CAP}(e) \). 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 \( P(j) \) evacuees who must reach the anti-root safe node \( \text{SAFE} \) while following the arcs of related path \( \Gamma(j) \). The last \( j\)-evacuee must reach \( \text{SAFE} \) before deadline \( \Delta_j \). Only one arc \( e(j) \) has origin \( j \) and only one arc has destination \( \text{SAFE} \).

- 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 \( \text{Length}(j) \). Every arc \( e \in E \) is also provided with some capacity \( \text{CAP}(e) \): no more than \( \text{CAP}(e) \) 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 \( T^*_j \) (when a first \( j\)-evacuee leaves \( j \)), its ending time \( T^0_j \), (when the last \( j\)-evacuee arrives to \( \text{SAFE} \)) and its evacuation rate \( v_j \), subject to (Evacuation Rate Formula): \( T^*_j = T^0_j + \text{Length}(j) + \text{P}(j) + v_j \).

Then the **Late Evacuation Problem (LEP)** consists in the search \( T^*_j \), \( T^0_j \), \( v_j \), \( j \in J \), consistent with deadlines and capacities, and maximizing the global safety margin \( \text{Inf} \{ \Delta_j - T^0_j \} \).

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

![Figure 1: An Instance of Tree-LEP](image)

**Figure 2:** Tree-LEP Schedule in RCPSP format.

In order to turn a Tree-LEP solution into RCPSP format, we set, for any \( j \in J \): \( R_j = \text{Length}(j) \) and \( v_{\text{min}} = \text{P}(j) + (\Delta_j - R_j) \). Deadline constraint implies \( v_j \geq v_{\text{min}} \); \( \text{vmax} = \text{CAP}(e(j)) \). Then we consider the process defined by the \( j\)-evacuees when they enter into the \( \text{SAFE} \) node, and call it evacuation job \( j \). Its starting time is \( T_j = T^0_j + \text{Length}(j) \), its ending time is \( T^*_j = T^0_j + \text{Length}(j) \) and we want to maximize \( \text{Safe-Margin} = \text{Min} \{ j \in J (\Delta_j - T^*_j) \} \). If \( v_j \) denotes related evacuation rate, we get the following **temporal** constraints: \( R_j \leq T_j \leq T^0_j \leq \Delta_j \) and \( T^*_j = T_j + \text{P}(j) + v_j \). As for resource constraints, we say that 2 jobs \( j_1, j_2 \), overlap iff interval \( [T_{j_1}, T^*_{j_1}] \cap [T_{j_2}, T^*_{j_2}] \) is neither empty nor reduced to one point. Then **resource** constraints tells that for any arc \( e \) in \( A \) and for any **Overlap clique** \( J_0 \subseteq J(e) \) = \( \{ j \text{ such that } e \in \Gamma(j) \} \), we should have: \( \Sigma j \in J \cap J(e) v_j \leq \text{CAP}(e) \). In case \( J_0 = e(j) \), this yields \( v_j \leq v_{\text{max}} \).

### 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 \( \Delta_j \), maximal evacuation rate \( \text{vmax} \) and set subset \( \Gamma(j) \subseteq E \) of resources used by \( j \); for any \( e \in E \), the Capacity \( \text{CAP}(e) \) 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-RCPS Model: Compute Rational Vectors

\[ T = (T_j, j = 1..N), T^* = (T^*_j, j = 1..N), v = (v_j, j = 1..N) \geq 0, \text{ and } \{0, 1, \cdot \cdot \cdot \} \text{-valued vector } \Pi = (\Pi_{j1,j2}, j, j \in 1..N) \]

Semantic: \( \Pi_{j1,j2} = 1 \Leftrightarrow j_1 < j_2; \Pi_{j1,j2} = 0 \Leftrightarrow j_1 > j_2 \). 

Overlap 2, such that:

- **Structural Constraints:** For any \( j_1, j_2 \), 
  \[ \Pi_{j1,j2} = - \Pi_{j2,j1} \]
- **Temporal Constraints:**
  - For any \( j \): \( R_j \leq T_j \leq T^*_j \leq \Delta_j \)
  - 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 \( j \): \( v_j = P(j)(\Delta_j - R_j) \)
  - For any arc \( e \), \( E2 \) holds: \( (J_0 \leq J) \) is such that for any pair \( j_1, j_2 \) in \( J_0 \), 
    \[ \Pi_{j1,j2} = 0 \Rightarrow \sum_{j \in J_0 \cap j_2} v_j \leq \text{CAP}(e); \]

- **Maximize:** Safe-Margin = \( \text{Inf}_{j}(\Delta_j - T^*_j) \)

This model fits with industrial contexts, where jobs \( j \) involving continuous flows of items are applied a model.

For any \( j \), \( T^*_j = T_j + P(j)\nu_j \); \( E1^* \)

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

**Linear-MSM-RCPS-Cut(1):**

Initialization a set \( W \) of constraints (E1, E1*) and consider related restriction MSM-RCPS(\{1, W\}); Not Stop;

While Not Stop do

Solve LINEAR-MSM-RCPS(\{1, W\});

Search for \( j_0 \) \( (j_1, j_2) \) and \( w_0 \) such that (E1, E1*) do not hold;

If Fail(\text{Search}) then Stop
Else Insert (E1, E1*) related to \( j_0, w_0 \) into \( W \).

**4 PREEMPTIVE MSM-RCPS**

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

**Figure 3:** P-MSM-RCPS 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 \( \{t_1, \cdot \cdot \cdot , t_{2N}\} \), in such a way that \( t_1 \leq t_2 \leq \cdot \cdot \cdot \leq t_{2N}. \) For any \( k \in 1, \cdot \cdot \cdot , 2N-1 \), we set \( \delta_k = t_{k+1} - t_k. \) This leads to the following PL Preemptive(\( \lambda \)):

**Preemptive(\( \lambda \)) Linear Program:** Compute rational vector \( w = (w_{j,k}, j \in K, k = 1..2N-1) \geq 0, \) 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 \( j, k, \) \( w_{j,k} \leq v_{j,k} \);
- For any \( j, \sum_{k} \delta_k w_{j,k} = P(j) \);
For any arc \( e \), any \( k \): \( \sum_{j \in E} w_{j,k} \leq \text{CAP}(e) \);

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(\( \lambda \)) identifies a preemptive schedule which is consistent with safety margin \( \lambda \), in case such a schedule exists.

Proof: If a preemptive schedule exists, consistent with safety margin \( \lambda \), release dates \( R_{j} \), deadlines \( \Delta_{j} \), \( j \in J \), and capacities \( \text{CAP}(e), e \in A \), then it can be chosen in such a way that for any \( \lambda \) 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 \( \lambda \)-Val by making \( \lambda \) iteratively evolve between a non feasible value \( \lambda \) and a feasible one \( \lambda_0 \):

Optimal-P-MSM-RCPSP(Threshold):

\( \lambda_{0} \leftarrow 0 ; \lambda_{1} \leftarrow \text{Inf}_{j} [\Delta(j) - (R_{j} + P(j)v_{\max})] ; w_{-Sol} \leftarrow \text{Nil} ; \lambda_{Val} \leftarrow - \infty ; \text{Solve Preemptive}(\lambda_{1}) ; \)

If Success(Solve) then \( \lambda_{Val} \leftarrow \lambda_{1} ; w_{-Sol} \leftarrow \text{related vector} \)

Else

\( \text{Solve Preemptive}(\lambda_{0}) ; \)

If Success(Solve) then

\( \lambda_{Val} \leftarrow \lambda_{0} ; w_{-Sol} \leftarrow \text{related vector} \); Counter \( \leftarrow 0 ; \)

While Counter \( \leq \) Threshold do

\( \lambda \leftarrow (\lambda_{1} + \lambda_{0}) / 2 ; \text{Solve Preemptive}(\lambda) ; \)

If Success(Solve) then \( \lambda_{0} \leftarrow \lambda ; \lambda_{Val} \leftarrow \lambda \); w_{-Sol} \leftarrow \text{related w} ; \)

Else \( \lambda_{1} \leftarrow \lambda ; \text{Optimal-P-MSM-RCPSP} \leftarrow (\lambda_{Val} - w_{-Sol}) ; \)

Else Optimal-P-MSM-RCPSP \( \leftarrow \) 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(\( \lambda \)). As for complexity, we set Threshold = \( \text{Log}_{2}(\text{Maximal binary encoding size of } \Delta_{j} \) 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:

- Sterilization 1: Smoothing w while keeping safety margin \( \lambda \) as in Figure 4 below:

- Sterilization 2: Deriving from w a topological vector \( \Pi \), and solving MSM-RCPSP(\( \Pi \)).

5 A FLOW BASED HEURISTIC

This section is devoted to the description of a 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 \( T^{e}_{j} - T^{e}_{j} \) and height is the evacuation rate \( v_{j} \), then, if \( j \) 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 \( v_{j} \) 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 \( w = (w_{j_{1}j_{2}}) \) \( j_{1} \), \( j_{2} \in J(e) \cup \{s, p\} \geq 0 \), which represents, for all resources \( e \), the way jobs share resource \( e \). Clearly, this vector \( w \) must satisfy standard flow conservation laws:

- For any \( e \) : \( \Sigma_{j_{1} \cup j_{2}} w_{j_{1}j_{2}} = \Sigma_{j_{1} \cup j_{2}} w_{j_{1}j_{2}} = \text{CAP}(e) \);

- For any resource \( e \) of \( F \) and any job \( j_{0} \in J(e) \), \( \Sigma_{j_{1} \cup j_{2}} w_{j_{1}j_{2}} = \Sigma_{j_{1} \cup j_{2}} w_{j_{1}j_{2}} = v_{j_{0}} \).

Besides, if we introduce starting times \( T^{e}_{j} \) and ending times \( T^{e}_{j} \) as in II, then, for any \( j_{1}, j_{2} \), the following implication is true: \( \Sigma_{e} w_{j_{1}j_{2}} \neq 0 \Rightarrow T^{e}_{j_{2}} \geq T^{e}_{j_{1}} \). This logical constraint means that if job \( j_{1} \) provides \( j_{2} \) with some part of resource \( e \), then \( j_{2} \) should be achieved before \( j_{2} \) starts. Clearly, we must keep on with the other standard constraints:

- For any \( j \) : \( R_{j} \leq T^{e}_{j} \leq \Delta_{j} ; v_{j} \leq v_{\max} ; T^{e}_{j} = T_{j} + P(j)v_{j} ; T_{j} = T^{e}_{j} ; T_{j} = 0 \).

- Maximize \( \min(\Delta_{j} - T^{e}_{j}) \).
5.1 An Adaptive 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 \( T(j) \). In order to find a compromise we control an adaptive safety margin \( \lambda \), through binary search and a related priority list \( \sigma \), which drives the insertion process for a given \( \lambda \). For a given value of \( \lambda \), and a current list \( \sigma \), the procedure \( \text{Insert-MSM-RCPSP}(\lambda, \sigma) \) scans the jobs \( j_0 \) in \( \sigma \), and try to compute \( T_0 \) and \( v_0 \) in such a way that MSM-RCPSP-Flow constraints are satisfied for all jobs \( j \) before or equal to \( j_0 \) according to \( \sigma \), and that \( v_0 \) is minimal. In case of success, then \( \lambda \) is increased, else \( \text{Insert-MSM-RCPSP}(\lambda, \sigma) \) yields a set of pairs \( j_1, j_2 \), asked to become such that \( j_1, j_2 \) (Instruction Update(\( \sigma \)) below).

\[
\text{MSM-RCPSP-Flow(Precision: Number)}
\]

Algorithmic Scheme:

**Step 1:** Start from a non feasible margin \( \lambda_{-\text{max}} \), a feasible one \( \lambda_{-\text{min}} \), a related Current-Schedule; Initialize priority list \( \sigma \): priority given to jobs \( j \) with small \( P(j) \) and expected safety.

While \((\lambda_{-\text{max}} - \lambda_{-\text{min}}) \geq \text{Precision}\)

\[
\lambda \leftarrow (\lambda_{-\text{min}} + \lambda_{-\text{max}}) / 2; \text{ Insert-MSM}(\lambda, \sigma)
\]

If Success then set \( \lambda_{-\text{min}} \) to \( \lambda \), and Update Current-Schedule.

Else Update(\( \sigma \)); Retrieve topology \( \Pi \) from Current-Schedule.

**Step2:** Solve resulting P-MSM-RCPSP(\( \Pi \)).

5.2 Insert-MSM Procedure

This procedure works while scanning current priority list \( \sigma \) and assigning \( T_j \) and \( v_j \) values as far as jobs \( j \) come. That means that at some time during the process, we are considering some job \( j_0 \), while all jobs \( j \) such that \( j \neq j_0 \) have been scheduled: for any \( j \in J \cup \{s\} \) such that \( j \neq j_0 \) and \( j \neq j_0 \), we are provided with values \( T_j, T^*, v_j \), as well as with values \( \Phi(e, j) \) which represent 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:

1. **1st step**: Scan \( I(j_0) \) according to decreasing \( \Phi(e, j_0) \) values, and for any \( e \in I(j_0) \), provide \( j_0 \) with an amount of resource \( e \) in such a way resulting \( T^*_{j_0} \) does not exceed \( \Delta_0 - \lambda \).

2. **2nd step**: In case of success of previous first step, we become provided with an evacuation rate \( v_{j_0} \) and, for any resource \( e \neq e_0 \) in \( I(j_0) \) with an evacuation rate value \( v_{e-\text{aux}} \) which may be less than \( v_{j_0} \). So the second step makes increase the values \( w^e_{j_0} \) 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 \( I(j_0) \).

3. **3rd step**: 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 \( w^e_{j_0} \) values, and works by shifting, as far as possible, values \( w^e_{j_0} \) which involve, for a given \( j_0 \), only one resource \( e \), to another job \( j' \) such that \( j' \in J(e) \), \( w^{e'}_{j_0} \neq 0 \) and \( \Pi'(e, j', \text{Index}) \geq w^{e'}_{j_0} + w^e_{j_0} \).

Example 2: Suppose that we face here the following situation: \( \sigma = s, j_1, j_2, ... j_3, ... j_6; I(j_0) = \{e_1, e_2\}; \text{CAP}(e_1) = 20, \text{CAP}(e_2) = 25; \Delta_0 = 21; P(j_0) = 5; R_1 = 10; j_1 \in J(e_1) \cap J(e_2); j_2 \in J(e_1); j_3 \in J(e_3); P(j_1) = 6; P(j_2) = 3; P(j_3) = 4. \)

Then we get:

**Step1** \(- w^{e_1}_{j_0} = 2; w^{e_2}_{j_0} = 3; w^{e_1}_{j_0} = 8; \quad \Rightarrow \quad v_{j_0} = 10; \text{Success}.**

**Step2** \(- w^{e_2}_{j_0} = 7; \text{Success}; **Step3** \(- w^{e_1}_{j_0} = 0; w^{e_2}_{j_0} = 8; T_{j_0} = 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 \( U = (U_j, j \in J(s)) \), \( U^n = (U^n_j, j \in J(s)) \) such that:
\[ J(s) \] denotes the set of jobs \( j \) such that \( U_j \) and \( U^*_j \) are defined;
\[ \text{If } f_j \in J(s), \text{ then } A_j < U_j < U^*_j \leq B_j. \]

For a given job \( j \), the meaning of \( U_j \) and \( U^*_j \) is that \( w_j = w_j(t) \) must be constant on \([U_j, U^*_j]\) and such that, for any \( t \) outside \([U_j, U^*_j]\), \( w_j(t) \) inside \([U_j, U^*_j]\). Then Branching from \( s \). Given a job \( j \) and 2 values \( \alpha \) and \( \beta \) such that \( A_j < \alpha < \beta \), node \( s \) gives rise to 3 sons:
- **First son**: \( A_j \) is replaced by \( \alpha \);
- **Second son**: \( B_j \) is replaced by \( \beta \);
- **Third son**: \( U_j \) is replaced by \( \alpha \) and \( U^*_j \) by \( \beta \);
we must have: \( \alpha < U_j < U^*_j < \beta \).

The 3-uple \((i, \alpha, \beta)\) 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 \( \lambda_-\text{Val} \) and related Branching Signature \( \text{Sign} = (j_0, a_0, b_0) \).

**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 \( B^*_j = \text{Inf}(B_j, A_j - \lambda) \) and \( S = \{A_j, B^*_j, j \in J\} \cup \{U_j, U^*_j, j \in J(s)\} \). We order \( S = \{t_1, \ldots, t_6\} \) through increasing values \( t_1 < t_2 < \ldots < t_6 \) and set, for any \( k = 1..K-1 \): \( \delta_k = t_{k+1} - t_k \)
- We build 4 vectors \( k1, k2, k3, k4 \), with indexation on \( J \), and whose meaning is:
  - \( k_1 \): means the value \( j \) such that \( A_j = t_1 \);
  - \( k_2 \): means the value \( j \) such that \( B^*_j = t_1 \);
  - \( k_3 \): means the value \( j \) such that \( U_j = t_1 \); (** If \( U_j \) is undefined, then \( k_3 \) is \( 0^* \))
  - \( k_4 \): means the value \( j \) such that \( U^*_j = t_1 \); (** If \( U^*_j \) is undefined, then \( k_4 \) is \( 0^* \))

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

**Preemptive**\((\lambda)\): (Compute \( w = w_{j,k}, j \in J, k = 1..K-1 \) such that;
- For any \( e \) and any \( k \), \( \Sigma_{j \in J(e)} w_{j,k} \leq \text{CAP}(e) \)
- For any \( j, \Sigma_{i \in \text{DI}_j} w_{j,k} = \text{P}(j) \)
- For any \( j, \text{any } k \leq k1(j) - 1, w_{j,k} = 0 \)
- For any \( j, \text{any } k \leq k2(j), w_{j,k} = 0 \)
- For any \( j, \text{any } k \leq k3(j), w_{j,k} \leq w_{j,k} \)
- For any \( j, \text{any } k \leq k4(j) - 2, w_{j,k} \leq w_{j,k} \).

We try to turn a solution of **Preemptive**\((\lambda)\) into a MSM-RCPSp Solution through procedures

**Sterilization**, \( x = 1, 2 \) 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 \( \lambda_-\text{Val} = UB(s) \).

**Branching Strategy**: Let us suppose that we just computed \( \lambda_-\text{Val} = UB(s) \), 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 \( \Sigma' = \{\Sigma_1, \ldots, \Sigma_K\} \) representative of the resource profile induced by \( j \):
- **1**th Configuration: A hole (see Fig. 6) with some depth and width and a weight = depth.width;
- **2**nd 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.](image)

So our Branching Strategy comes as follows: In case Configuration 1, then we compute branching signature \( \text{Sign} \) 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.

<table>
<thead>
<tr>
<th>Instances</th>
<th>Nodes Cap-Relax</th>
<th>Congest</th>
</tr>
</thead>
<tbody>
<tr>
<td>dense_10</td>
<td>19.80</td>
<td>155.06</td>
</tr>
<tr>
<td>dense_15</td>
<td>29.10</td>
<td>160.08</td>
</tr>
<tr>
<td>dense_20</td>
<td>38.60</td>
<td>164.88</td>
</tr>
<tr>
<td>medium_10</td>
<td>19.70</td>
<td>152.83</td>
</tr>
<tr>
<td>medium_15</td>
<td>29.10</td>
<td>159.39</td>
</tr>
<tr>
<td>medium_20</td>
<td>38.20</td>
<td>160.69</td>
</tr>
<tr>
<td>medium_25</td>
<td>46.80</td>
<td>169.91</td>
</tr>
<tr>
<td>sparse_10</td>
<td>19.50</td>
<td>146.17</td>
</tr>
<tr>
<td>sparse_15</td>
<td>28.80</td>
<td>153.92</td>
</tr>
<tr>
<td>sparse_20</td>
<td>38.30</td>
<td>157.78</td>
</tr>
<tr>
<td>sparse_25</td>
<td>47.60</td>
<td>154.73</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Instances</th>
<th>Opt-P-MSM</th>
<th>Opt-P-CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>dense_10</td>
<td>106.25</td>
<td>0.05</td>
</tr>
<tr>
<td>dense_15</td>
<td>68.3</td>
<td>0.08</td>
</tr>
<tr>
<td>dense_20</td>
<td>39.29</td>
<td>0.11</td>
</tr>
<tr>
<td>medium_10</td>
<td>92.59</td>
<td>0.04</td>
</tr>
<tr>
<td>medium_15</td>
<td>65.35</td>
<td>0.08</td>
</tr>
<tr>
<td>medium_20</td>
<td>54.85</td>
<td>0.11</td>
</tr>
<tr>
<td>sparse_10</td>
<td>113.78</td>
<td>0.04</td>
</tr>
<tr>
<td>sparse_15</td>
<td>78.33</td>
<td>0.06</td>
</tr>
<tr>
<td>sparse_20</td>
<td>64.45</td>
<td>0.11</td>
</tr>
<tr>
<td>sparse_25</td>
<td>21.69</td>
<td>0.45</td>
</tr>
</tbody>
</table>

Table 2-Bis: Behavior of MSM-RCPSP-Flow.

<table>
<thead>
<tr>
<th>Instances</th>
<th>MSM-Flow</th>
<th>Preempt-Gap</th>
<th>Flow-CPU</th>
<th># fails</th>
</tr>
</thead>
<tbody>
<tr>
<td>dense_10</td>
<td>97.09</td>
<td>2.01</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>dense_15</td>
<td>58.02</td>
<td>2.31</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>dense_20</td>
<td>34.75</td>
<td>2.97</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>medium_10</td>
<td>88.76</td>
<td>2.06</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>medium_15</td>
<td>52.87</td>
<td>2.82</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>medium_20</td>
<td>43.75</td>
<td>3.53</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>medium_25</td>
<td>36.78</td>
<td>1.96</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>sparse_10</td>
<td>110.38</td>
<td>1.88</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>sparse_15</td>
<td>75.77</td>
<td>2.75</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>sparse_20</td>
<td>48.70</td>
<td>3.94</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>sparse_25</td>
<td>32.67</td>
<td>*</td>
<td>1.18</td>
<td>4</td>
</tr>
</tbody>
</table>

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 of nodes of the search tree which were visited during the process.

Table 3: Behavior of B&B-MSM-RCPSP.

<table>
<thead>
<tr>
<th>Instances</th>
<th>Opt-P-MSM</th>
<th>B&amp;B-MSM-Inf</th>
<th>B&amp;B-MSM-Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>dense_10</td>
<td>106.25</td>
<td>105.75</td>
<td>105.76</td>
</tr>
<tr>
<td>dense_15</td>
<td>68.30</td>
<td>66.65</td>
<td>67.49</td>
</tr>
<tr>
<td>dense_20</td>
<td>39.29</td>
<td>38.86</td>
<td>39.29</td>
</tr>
<tr>
<td>medium_10</td>
<td>92.58</td>
<td>91.87</td>
<td>91.87</td>
</tr>
<tr>
<td>medium_15</td>
<td>65.35</td>
<td>63.42</td>
<td>64.48</td>
</tr>
<tr>
<td>medium_20</td>
<td>54.85</td>
<td>49.82</td>
<td>54.57</td>
</tr>
<tr>
<td>medium_25</td>
<td>51.60</td>
<td>49.19</td>
<td>51.60</td>
</tr>
<tr>
<td>sparse_10</td>
<td>113.78</td>
<td>113.75</td>
<td>113.78</td>
</tr>
<tr>
<td>sparse_15</td>
<td>78.33</td>
<td>78.33</td>
<td>78.33</td>
</tr>
<tr>
<td>sparse_20</td>
<td>64.45</td>
<td>59.21</td>
<td>64.38</td>
</tr>
<tr>
<td>sparse_25</td>
<td>34.34</td>
<td>32.49</td>
<td>34.34</td>
</tr>
</tbody>
</table>
Table 3-Bis: Behavior of B&B-MSM-RCPS

<table>
<thead>
<tr>
<th>Instances</th>
<th>Nodes</th>
<th>B&amp;B-CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>dense_10</td>
<td>12077.80</td>
<td>361.14</td>
</tr>
<tr>
<td>dense_15</td>
<td>27148.80</td>
<td>1090.40</td>
</tr>
<tr>
<td>dense_20</td>
<td>11338.90</td>
<td>720.85</td>
</tr>
<tr>
<td>medium_10</td>
<td>4253.30</td>
<td>105.51</td>
</tr>
<tr>
<td>medium_15</td>
<td>28850.20</td>
<td>1440.62</td>
</tr>
<tr>
<td>medium_20</td>
<td>27254.90</td>
<td>1800.03</td>
</tr>
<tr>
<td>medium_25</td>
<td>10839.00</td>
<td>1081.33</td>
</tr>
<tr>
<td>sparse_10</td>
<td>57425.50</td>
<td>604.36</td>
</tr>
<tr>
<td>sparse_15</td>
<td>10668.30</td>
<td>360.32</td>
</tr>
<tr>
<td>sparse_20</td>
<td>15164.50</td>
<td>1440.08</td>
</tr>
<tr>
<td>sparse_25</td>
<td>17054.10</td>
<td>1800.28</td>
</tr>
</tbody>
</table>

Comment: B&B-MSM-Inf is always very close to optimistic estimation Opt-P-MSM, and Optimal-P-MSM-RCPS 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 preemptive 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


