Proceedings of the Federated Conference on Computer Science and Information Systems pp. 143–146 DOI: 10.15439/2019F90 ISSN 2300-5963 ACSIS, Vol. 18

# Models and Algorithms for Natural Disaster Evacuation Problems

Christian Artigues
LAAS CNRS
TOULOUSE, France
Email: artigues@laas.fr

Emmanuel Hebrard
LAAS CNRS
TOULOUSE, France
Email: hebrard@laas.fr

Alain Quilliot
LIMOS CNRS UMR 6158
LABEX IMOBS3, Université Clermont-Auvergne
Bat. ISIMA, BP 10125
Campus des Cézaux,
63173 Aubière, France
Email: quilliot@isima.fr

Hélène Toussaint
LIMOS CNRS UMR 6158
LABEX IMOBS3, CNRS

Bat. ISIMA, BP 10125
Campus des Cézaux,
63173 Aubière, France
Email: toussain@isima.fr

***Abstract*— We deal here, in the context of a H2020 project, with the design of evacuation plans in face of natural disasters: wildfire, flooding… People and goods have to been transferred from endangered places to safe places. So we schedule evacuee moves along pre-computed paths while respecting arc capacities and deadlines. We model this scheduling problem as a kind of multi-mode *Resource Constrained Project Scheduling* problem (RCPSP) and handle it through network flow techniques.**

## I. Introduction

THIS work has been carried on in the context of the H2020 GEOSAFE European project [4], whose overall objective is 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 a natural disaster, mainly a wildfire or a flooding. In such a circumstance, decisions which have to be taken are about fighting the cause of the disaster, adapting standard logistics (food, drinkable water, health…) to the current state of infrastructures, and evacuating endangered areas (see [2]). We focus here on the *late evacuation problem*, that means the evacuation of people and eventually critical goods which have been staying at their place as long as possible.

While evaluation planning remains mostly designed by experts, 2-step optimization approaches have been addressed [2]: 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. As a matter of fact, this last step involves 2 distinct work pieces, one about forecasting, difficult in the case of wildfire, because of their dependence to topography and meteorology [4], and the second one about priority rules and evacuation rates imposed to evacuees [3]. The model which we study here is closed to the one proposed in [1] and called the *non preemptive evacuation planning problem* (NEPP). According to it, remaining evacuees have been clustered into groups with same original location and pre-computed route, and once a group starts moving, then it must keep on at the same rate until reaching his target safe area (*Non Preemption* hypothesis, which matches practical concerns of the people who supervise the evacuation process). While authors in [1] address their model while discretizing both the time space and the rate domains and applying constraint propagation techniques, we consider it as an extension of the *Resource Constrained Project Scheduling Problem* (RCPSP: [5,6]), with continuous variables which identify evacuation rates and with an objective function which reflects the safety provided to every evacuee. We use this RCPSP reformulation in order to design a heuristic algorithm which deals with our problem according to network flow like techniques, well-fitted to real-time emergency contexts.

The paper is structured as follows: Section 2 provides the NEPP model. Section 3 describes our RCPSP reformulation. Sections 4, 5 are about algorithms and numerical tests.

## II. Non Preemptive Evacuation Planning (NPEP)

We consider here a transit network $H = (N, A)$: $N$ is its node set and $A$ its arc set; Every arc $e \in A$ is provided with the time $TIME(e)$ required for some evacuee to move through $e$ and with the maximum number $CAP(e)$ of evacuees who may engage themselves $e$ per time unit. We distinguish:

- The *Evacuation* node subset $N^+$, whose nodes are labelled $i = 1..n$ and related to some population $P(i)$.
- The *Safe* node subset $N^-$ and the *Relay* node subset $N^=$.

Evacuees of the population $P(i)$ located at $i \in N^+$ move along a pre-determined path $\Gamma(i)$, that means a sequence of arcs $e^i_1,.., e^i_{k(i)}$ connecting $i$ to some *safe* node $S(i)$. We set $L\_TIME(i) = \Sigma_{k = 1..k(i)} TIME(e^i_k)$, and, for any $k =$

$1..k(i)$: $L(i, k) = \Sigma_{k \leq j} TIME(e^i_k)$ and $L^*(i, k) = \Sigma_{k \geq j} TIME(e^i_k)$.

We must comply with capacity restrictions: During one time unit, no more than $Deb(i)$ evacuees may start moving from $i \in N^+$ and no more than $CAP(e)$ evacuees may simultaneously engage themselves on a given arc $e$. Also, forecast about the way the natural disaster will evolve imposes that for any arc $e$ of the transit network, nobody may start moving along $e$ after deadline $Dead(e)$, while the whole evacuation process should be over at global deadline $T\text{-}Max$. Thus all evacuees coming from $i \in N^+$ should reach related *safe* node $S(i)$, before $\Delta(i) = \text{Inf}\,(T\text{-}Max, \text{Inf}_{k = 1..k(i)} (Dead(e^i_k) + L^*(i, k))$.

Besides, authorities impose *Non Preemption* : once evacuees related to *evacuation* node $i$ have started moving, they must keep on at the same speed and rate along path $\Gamma(i)$, until they all reach *safe* node $S(i)$. We denote by $v_i$ the related *evacuation rate* (number of evacuees per time unit which enter on $\Gamma(i)$ at until $i$ becomes empty. We derive an upper bound $v\text{-}max(i)$ for $v_i$ by setting: $v\text{-}max(i) = \text{Inf}\,(\text{Inf}_j\, CAP(e^i_k)), Deb(i))$. We also see that if we are provided with the *start-date* $T_i$ of $i$ evacuation process and with its *evacuation rate* $v_i$ then we deduce its *end-date* $T^*_i = T_i + L\_TIME(i) + P(i)/v_i$. We deduce from deadline $\Delta(i)$ a minimal evacuation rate $v\text{-}min(i) = P(i)/(\Delta(i) - L\_TIME(i))$.

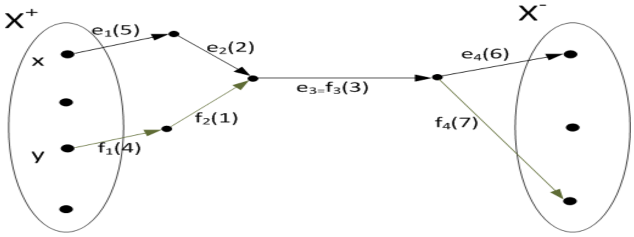
Then, the *Non Preemptive Evacuation Planning Problem* (NEPP) is about the computation of an *evacuation schedule*, which means of *start-times* $T_i$ and *evacuation rates* $v_i$, $i \in N+$. The quality of such a *schedule* $\Lambda = (T,v)$ is going to be the *weighted safety margin* $\Sigma_i P(i).(\Delta(i) - T^*_i)$.

## III. A RCPSP Oriented Reformulation of NEPP.

We identify evacuation nodes $i$ of network $H$ and related *evacuation jobs*. So the key idea here is to consider the arcs $e$ of the network $H$ as resources, likely to be exchanged by *evacuation jobs* $i$, $j$ whose paths $\Gamma(i)$ and $\Gamma(j)$ share arc $e$. In order to formalize it, we introduce *Conditional Time Lags*:

- If $\Gamma(i) = \{e^i_1,.., e^i_{k(i)}\}$ and $\Gamma(j) = \{f^j_1,..,f^j_{k(j)}\}$ share arc $e = e^i_k = f^j_l$, and if evacuees from $j$ come on $e$ after evacuees from $i$, then delay $T_j - T_i$ will be no smaller than $TL\text{-}Elem(i, j, e) = L(i, k\text{-}1) - L(j, l\text{-}1) + P(i)/v_i$.
- Set $Arc(i,j) = \{e \in \Gamma(i) \cap \Gamma(j)\}$ and $TL(i,j, v_i) = \text{Sup}_{e \in Arc(i,j)} (L(i, k\text{-}1) - L(j, l\text{-}1) + P(i)/v_i) =$ *Conditional Time Lag* between $i$ and $j$. If $Arc(i,j) \neq$ Nil and evacuees of $j$ enter after evacuees of $i$ on the arcs of $Arc(i,j)$, then we must have $T_j \geq T_i + TL(i,j, v_i)$. We notice $TL(i,j, v_i)$ depends in a convex way on the *evacuation rate* $v_i$ of $i$.

This notion is illustrated by following Figure 1:




$P(x) = 50$, $v_y = 10$; $P(y) = 40$, $v_y = 15$ =>
$L(x,2) = 7$; $L(y, 2) = 5$; $TL(x,y\ v_x) = 7 + 4 - 5 = 6$.

**Figure 1**: *Conditional Time Lags*.

We derive a RCPSP (*Resource Constrained Scheduling*: [5,6]) reformulation of NEPP, which relies on the fact that we consider every *evacuation job* $i \in N^+$ as a job, whose execution requires resources which are arcs $e \in \Gamma(i)$, constrained by their capacities $CAP(e)$ and whose *start-dates* are constrained by *conditional time lags*:

***NPEP-RCPSP*** Model :

{*Preliminary* : We add to the set $N^+$ two fictitious jobs $s$ (*source*) and $p$ (*sink*), in order to express the way resources are exchanged between jobs as a flow vector. Then we set, for any $i \in N^+$: $TL(s,i, CAP(e)) = 0$ and $TL(i,p, v_i) = L\_TIME(i) + P(i)/v_i$.

*Output Vectors* : For any $i$ in $N^+ \cup \{s,p\}$ compute *start-date* $T_i$ and *evacuation rate* $v_i$; In order to do it we involve, for any pair $(i,j)$ and any arc $e$ in $Arc(i,j)$ the part $w_{i,j,e}$ of access rate to $e$ which is given by $i$ to $j$

*Constraints* :

- For any $i \neq p$, $T_i + L\_TIME(i) + P(i)/ v_i \leq \Delta(i)$ ; (**Deadline Constraints**) (E1)
- for any pair $(i,j)$ and any $e$ in $Arc(i,j)$, $w_{i,j,e} \neq 0$ -> $T_j \geq T_i + TL(i,j, v_i)$; (**Conditional Time Lag Constraints**) (E2)
- $T_s = 0$ ; (E3)
- For any $i$ in $N^+$, $N^+$ and any arc $e$ in $\Gamma(i)$, (**Flow Constraints**): $\Sigma_{j \text{ such that } e \in Arc(x,y)} w_{i,j,e} = v_i = \Sigma_{j \text{ such that } e \in Arc(j,i)} w_{j,i,e}$; (E4)
- For any arc $e$ of the transit network $H$ : (**Flow Constraints**):$CAP(e) = \Sigma_{i \text{ such that } e \in \Gamma(i)} w_{s,i,e} = \Sigma_{i \text{ such that } e \in \Gamma(i)} w_{i,p,e}$; (E5)
- For any $i$ dans $N^+$, $v\text{-}Min(i) \leq v_i \leq v\text{-}Max(i)$. (E6)

*Maximize*: $\Sigma_i P(i).(\Delta(i) - T_i - L\_TIME(i) - P(i)/v_i)$

**Explanation**: (E1) tells that every *evacuation job* $i$ must be achieved before deadline $\Delta(i)$. (E2) means that if job $i$ provides $j$ with some access to arc $e$, then the *conditional time lag* inequality holds. (E4, E5) express Flow Kirshoff laws: arcs $e$ are resources that *evacuation jobs* exchange between them; so job $i$ receives $v_i$ resource (*evacuation rate*) for any $e \in \Gamma(i)$ and no more than $CAP(e)$ such resource may be simultaneously distributed between *evacuation jobs* .

## IV. ALGORITHMS

NMEP model contains both NP-Hard RCPSP and TSP problems. We have to choose between assigning high *rates* $v_i$ to *jobs* $i$ or let them monopolize the access to transit arcs, or conversely restricting $v_i$ in order to make $i$ share its arcs. In order to do it, we implement a two-step approach: *MNEP-First-Step* searches a feasible schedule satisfying (E1,..,E6), while *MNEP-Second-Step* increases *rates* $v_i$ in order to improve the *weighted safety margin*.

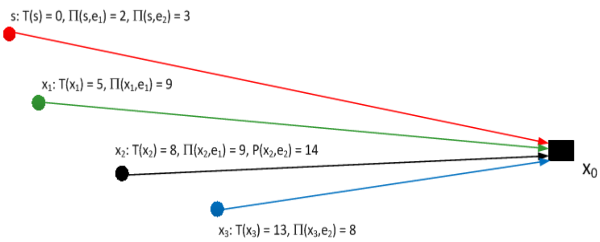
### A. The Greedy-NPEP Process.

*Greedy-NPEP* starts from some linear ordering σ defined on $N^+ \cup \{s,p\}$, and considers at any time some job $i_0$ such that for any $j$ prior to $i_0$ according to σ, $v_j$, $T_j$ and values $\Pi(j,e)$ = access level to arc $e$ that job $j$ can transmit to $i_0$ are available.Then it applies a 3 stage function *Assign*($i_0$) which computes (see Fig. 1) $v_{i0}$, $T_{i0}$ and flow values $w_{j,i0,e}$, $j$ s.t $j$ σ $i_0$, and $e \in Arc(j,i_0)$, or, in case of failure, a job *j-fail* σ $i_0$ considered as cause of the failure.

- **(1)** : *Assign* scans path $\Gamma(i_0)$, and for any $e$ in $\Gamma(i_0)$, provides $i_0$ with access rate to $e$ in such a way resulting *end-date* $T^*_{i0} \le \Delta(i_0)$. (see Fig. 2):

  ***Assign*1**
  For $e$ in $\Gamma(i_0)$ do
  Let *L-Job* = {$j$ s.t ($j$ σ $i_0$) AND ($e \in Arc(j, i_0)$ AND ($\Pi(j, e) \neq 0$), ordered according to increasing $T_j + TL(j, i_0, v_j)$ values}; $v$ <- 0 ; Not *Stop* ;
  While *L-Job* ≠ Nil AND Not *Stop* do
  If $T_j + TL(j, i_0, v_j) + L\_TIME(x_0) + P(i_0)/(v + \Pi(j, e)) \le \Delta(i_0)$ then
  Compute $w$ such that $T_j + TL(j, i_0, v(j)) + L\_TIME(x_0) + P(i_0)/(v + w) = \Delta(i_0)$;
  *Stop* ; $v$ <- $v + w$ ; $w_{j,i0,e}$ <- $w$ ;
  Else $v$ <- $\Pi(j,e) + v$; $w_{j,i0,e}$ <- $\Pi(x,e)$ ;
  If Not *Stop* then *Fail* : Choose *j-Fail* in *L-Job*
  Else *v-aux*($e$) <- $v$ ;
  If Not *Fail* then $V_{i0}$ <- $\mathrm{Sup}_e$ *v-aux*($e$); $e_0$ <- Arg Sup.

  σ = $s, \ldots, x_1, \ldots, x_2, \ldots x_3, \ldots, x_0, \ldots$
  $\Gamma(x_0) = \{e_1, e_2\}$; $CAP(e_1) = 20$, $CAP(e_2) = 25$; $\Delta(x_0)$; $P(x_0) = 50$; $L\_TIME(x_0) = 10$; $Arc(x_2, x_0) = \{e_1\}$; $Arc(x_1, x_0) = \{e_1, e_2\}$; $Arc(x_3, x_0) = \{e_2\}$; $TL(s, x_0) = 0$; $TL(x_1, x_0) = 6$; $TL(x_2, x_0) = 3$; $TL(x_3, x_0) = 4$;




=>
*Assign-1* -> $w_{s,x0,e1} = 2$; $w_{s,x0,e2} = 3$; $w_{x1,x0,e1} = 8$; $v_{x0} = 10$; *Success*; *Assign- 2* -> $w_{x2,x0,e2} = 7$; *Success*; *Assign-3* -> $w_{x1,x0,e1} = 0$; $w_{x2,x0,e1} = 8$; $T_{x0} = 21$.

**Figure 2**: *Assign Process.*

- **(2)** : *Assign1* computes $v_{i0}$ and, for any $e \neq e_0$ in $\Gamma(i_0)$ a value *v-aux*($e$) which may be less than $v_{i0}$; So *Assign2* increases the $w_{j,i0,e}$ for $e \neq e_0$ in order to make *job* $i_0$ run at the same *rate* for all arcs $e$ of $\Gamma(i_0)$. This part of the *Assign* process may induce a failure which *Assign2* assign to some job *j-Fail*.
- **(3)**: *Assign3* makes decrease the number of arcs provided with non null $w_{j,i0,e}$ values by shifting values $w_{j,i0,e}$ which involve, for a given $j$, only one arc $e$, to another job $j'$ such that $e \in Arc(j', i_0)$, $w_{j',i0,e} \neq 0$ and $\Pi(j', e) \ge w_{j,i0,e} + w_{j',i0,e}$.

Then *Greedy-NPEP* comes as follows:

***Greedy-RCPSP-TL***(σ) :
$T_s$ <- 0 ; For any arc $e$ do $\Pi(s, e)$ <- $CAP(e)$; Not *Stop*;
While (Not *Stop*) and σ no fully scanned do
Apply *Assign* to current $i_0$ and partial schedule;
If *Success*(*Assign*) then
For $e$ in $\Gamma(i_0)$ and $j$ s.t ($j$ σ $i_0$) ∧ ($e \in Arc(j,i_0)$)
do $\Pi(i_0, e)$ <- $v_{i0}$; $\Pi(j,e)$ <- $\Pi(j,e) - w_{j,i0,e}$;
Else *Stop* ; Return the pair (*j-Fail*, $i_0$).

### B. NPEP-First-Step

*Greedy-NPEP* may fail even in the case when a solution ($T$, $v$, $w$) exists. It raises the question of the way we deal with linear ordering σ.

- ***Initialization of σ***. For any $i$, we set $SME(i) = \Delta(i) - L\_TIME(i) - 2.P(i)/(v\text{-}max(i) + v\text{-}min(i))$, and compute σ by randomly sorting $N^+$ in such a way that if $P(i) < P(j)$ and $SME(i) < SME(j)$, then $i$ σ $j$.
- ***Making σ evolve***. In case of failure, *Greedy-NPEP* returns a pair (*j-Fail*, $i_0$), and this pair is inserted into a *Tabu* like set *FORBID* whose meaning is: If ($j$, $i$) is FORBID, then we should have ($i$ σ $j$).

So, global process *NPEP-First-Step* comes as follows:

**Procedure *NPEP-First-Step***(*Max-Iter*: Threshold)
Initialize σ as described above ; *FORBID* <- Nil ;
*Iter* <- 0 ; Not *Stop* ; *Success* <- 0 ;
While (*Iter* ≤ *Iter-Max*) AND (Not *Success*) do
Generate σ consistent with *FORBID* and Apply *Greedy-NPEP*; If *Failure* then Search a *failure responsible* (*j-Fail*, $i_0$) pair and put into *FORBID*.

### C. NPEP-Second-Step

In case *NPEP-First-Step* yields a feasible solution ($T$, $v$, $w$) *NPEP-Second-Step* improves it, by acting on rates $v_i$ in such a way time lags $L\_TIME(i) + P(i)/v_i$ decrease in an *ad hoc* way. Let us denote by *U-Active*, the set of pairs ($i$,$j$) which are allowed to support non null $w_{i,j,e}$ flow values. We notice that if *U-Active* is fixed, then resulting restriction of NPEP is a convex optimization problem defined on the ($v$,$w$) polyhedron defined by (E4, E5, E6).
So we fix *U-Active* according to the end of *NPEP-First-Step*, and deal with induced convex program:

- We derive from current $v$, $w$, values $T^*_i$, related critical paths, and values $\lambda = \lambda(i)$, $i \in N^+ \geq 0$, such that $\Sigma_i P(i)$. $T^*_i = \Sigma_i \lambda(i)/v_i + Constant$: Vector $Grad = (Grad_i = -\lambda(i)/v^2_i, i \in N^+)$ is a sub-gradient vector;
- Then we modify $v$ and $w$ according to (I1): $v$ <- $v + V$ ; $w$ <- $w + W$, with $V$ and $W$ s.t $V.Grad < 0$ and $v + V$ and $w + W$ comply with (E4, E5, E6) and computed by solving *Project-Grad* following linear program:

***Project-Grad***(*U-Active*, $v$, $w$, $\delta$, *Grad*) LP :
{Compute $V = (V_i, i \in N^+)$, and $W = (W_{i,j,e}, (i, j) \in$ *U-Active*, $e \in Arc(i,j)$) such that;

- $\forall (i,j, e)$, $w_{i,j,e} + W_{i,j,e} \geq 0$ ;
- $\forall i \neq s,p$, $e \in \Gamma(i)$, $\Sigma_j W_{i,j,e} = \Sigma_j W_{j,I,e} = V_i$ ;
- $\forall e$, $\Sigma_j W_{s,j,e} = \Sigma_j W_{j,p,e} = 0$ ;
- $\forall i \neq s,p$, $v\text{-}Min(i) \leq v_i + V_i \leq v\text{-}Max(i)$ ;
- $2.\delta \geq \Sigma_{i \neq s,p} V_i. Grad(i) \geq \delta$}

Then *NPEP-Second-Step* comes as follows:

**Procedure *NPEP-Second-Step***:
Let ($T$, $v$, $w$) be the feasible solution computed by *NPEP-First-Step* and $T^*$ related *end-date* vector;
Derive *U-Active*; Not *Stop* ; *Val* <- $\Sigma_i P(i). T^*_i$;
While Not *Stop* do
Compute $\delta$ and coefficients $\lambda(i)$, $i \in N$ ;
Solve *Project-Grad*(*U-Active*, $v$, $w$, $\delta$, *Grad*);
If no solution then *Stop* Else
Apply (I1), update $T_i$, $T^*_i$ and related critical paths; If *Val-Aux* = $\Sigma_i P(i).T_i$; If *Val-Aux* $\geq$ *Val* then *Stop*.

## V. NUMERICAL EXPERIMENTS.

**Purpose**: Algorithms were implemented on AMD Opteron 2.1GHz. Our goal was to evaluate the ability of *NPEP-First-Step* to deal with tight deadlines and the ability of *NPEP-Second-Step* to improve this solution.

**Instances/outputs**: An instance is a path collection {$\Gamma(i)$, $i \in N^+$}, given together with values $P(i)$, $\Delta(i)$ and $TIME(e^i_k)$. It is summarized by a 3-uple: ($n$, $m$, $\alpha$), where $n$ = Car($N^+$), $m$ = number of arc $e$, and $\alpha$ is as above. We both created our own instances and used an instance generator of [1].In order to get benchmarks, we generated *ad hoc* schedules ($T$, $v$) and derived deadlines $\Delta(i)$ which made us be provided with almost optimal solutions.

**Outputs**: For every 10 instance package, we compute:

- The number *Trial* of iterations on $\sigma$ necessary to get a feasible solution through *NPEP-First-Step*;
- The improvement margin (%) *IMPROVE* induced by *NPEP-Second-Step*;
- The gap between *NPEP* .and optimal value *VAL*

Table below provides results for $\alpha \in [1,2]$.

| Inst. 1: $n$ = 20, $m$ = 10 | *Trial* | *IMPROVE (%)* | *GAP (%)* | *CPU-NPEP* |
|---|---|---|---|---|
| $\alpha = 1.2$ | 22.30 | 13.8 | 4.7 | 40.4 |
| $\alpha = 1.5$ | 2.50 | 29.5 | 13.0 | 12.3 |
| $\alpha = 1.7$ | 1.39 | 40.8 | 17.7 | 8.1 |
| $\alpha = 2.0$ | 1.08 | 61.7 | 19.3 | 5.2 |
| Inst. 1: $n$ = 30, $m$ = 15 | | | | |
| $\alpha = 1.2$ | 40.6 | 14.6 | 5.6 | 70.5 |
| $\alpha = 1.5$ | 6.60 | 30.2 | 14.5 | 19.5 |
| $\alpha = 1.7$ | 2.05 | 42.3 | 19.1 | 12.0 |
| $\alpha = 2.0$ | 1.19 | 65.5 | 22.5 | 7.9 |

**Comment**: Tighting deadlines $\Delta(i)$ improve solutions.

## VI. CONCLUSION

We described here a two-step RCPSP oriented algorithm for the *NPEP* Problem. Remains now to deal with the design of an exact method for small instances and with an integrated computation of routes $\Gamma(i)$.

## REFERENCES


[1] C.Artigues, E.Hebrard, Y.Pencolé, A.Schutt, P.Stuckey: A study of evacuation planning for wildfires; *17 th Int. Workshop on Constraint Modelling/Reformulation,* Lille, France, (2018).

[2] V.Bayram : Optimization models for large scale network evacuation planning and management : a review ; *Surveys in O.R and Management*, (2016), DOI : 10.1016/j.sorms.2016.11.001.

[3] C.Even, V.Pillac, P.Van Hentenryk: Convergent plans for large scale evacuation; In *Proc. 29 th AAAI Conf. On Artificial Intelligence*, Austin, Texas, p 1121-1127, (2015).

[4] Geo-Safe-; *MSCA-RISE 2015 European Project –id 691161*. http://fseg.gre.ac.uk/fire/geo-safe.html. Accessed Jue 12, (2018).

[5] M.J. Orji, S. Wei. Project Scheduling Under Resource Constraints: A Recent Survey. *Inter. Journal of Engineering Research & Technology (IJERT)* Vol. 2 Issue 2, (2013)

[6] A.Quilliot, H.Toussaint: Flow Polyedra and RCPSP, *RAIRO-RO*, 46-04, p 379-409, (2012)