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 be 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 its 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 $N_e$, whose nodes are labelled $i = 1...n$ and related to some population $P(i)$.
- The Safe node subset $N_s$ and the Relay node subset $N_r$.

Evacuees of the population $P(i)$ located at $i \in N_e$ move along a pre-determined path $F(i)$, that means a sequence of arcs $e_1,..., e_{n_d}$ connecting $i$ to some safe node $S(i)$. We set $L\_TIME(i) = \sum_{k} TIME(e_i^k)$, and, for any $k =$
1..k(i): L(i, k) = \sum_{k \leq j} TIME(e_{i,j}) and L*(i, k) = \sum_{k \geq j} TIME(e_{i,j}). We must comply with capacity restrictions: During one time unit, no more than Dep(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-Max. Thus all evacuees coming from i \( \in N^* \) should reach related safe node S(i), before \( \Delta(i) = Inf(T-Max, Inf_{k \in M} (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 \( P(i) \), 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 \( P(i) \) at until i becomes empty. We derive an upper bound \( v_{max}(i) \) for \( v_i \), by setting: \( v_{max}(i) = Inf(I_{j}(Dep(i), Dep(j)) \). We also see that if we are provided with the \( \Delta(i) \) a minimal evacuation rate \( v_{min}(i) = P(i) / (\Delta(i) - L_{MAX}(i)) \). 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 \( A = (T, v) \) is going to be the weighted safety margin \( \Sigma_i P(i).(\Delta(i) - T_{MAX})(i) \).

III. A RCPSP Oriented Reformulation of NPEP.

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 \( P(i) \) and \( P(j) \) share arc e. In order to formalize it, we introduce Conditional Time Lags:

- If \( I(i) = \{ e_{1},..,e_{k(i)} \} \) and \( J(j) = \{ f_{1},..,f_{k(j)} \} \) share arc \( e = e_{i,j} = f_{i,j} \), and if evacuees from \( j \) come on \( e \) after evacuees from \( i \), then delay \( T_{i,j} - T_j \) will be no smaller than \( TL\text{-Elem}(i, j, e) = L(i, k-1) - L(j, l-1) + P(i)/v_i \).
- Set \( Arc(i,j) = \{ e \in I(i) \cap J(j) \} \) and \( TL(i,j, v_i) = Sup_{e \in Arc(i,j)}(L(i, k-1) - L(j, l-1) + P(i)/v_i) \) Conditional Time Lag between \( i \) and \( j \). If \( Arc(i,j) \neq \emptyset \) and any \( e \in Arc(i,j) \), then we must have \( T_j \geq T_{i,j} + TL(i,j, v_i) \). 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 Figure 1:

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 \( e \in I(i) \), 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 \( TL(i,p, v_i) = L_{MAX}(i) + P(i)/v_i \). Output Vectors: For any \( i \in N^* \cup \{s,p\} \) compute \( \Delta(i) \) 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 \( j \).

**Constraints:**

- For any \( i \neq p, T_i + L_{MAX}(i) + P(i)/v_i \leq \Delta(i) \); (*Deadline Constraints*)
- For any pair \( (i,j) \) and any \( e \in Arc(i,j) \), \( w_{i,j,e} \neq 0 \rightarrow T_j \geq T_i + TL(i,j, v_i) \); (*Conditional Time Lag Constraints*)
- \( T_i = 0 \); (E3)
- For any \( i \in N^*, N^* \) and any arc \( e \in I(i) \), (*Flow Constraints*): \( \Sigma_{j \mid e \in Arc(x,y)} w_{i,j,e} = v_i \)
- For any arc \( e \) of the transit network \( H \): (*Flow Constraints*):\( CAP(e) = \Sigma_{i \mid e \in I(i)} \Sigma_{j \mid e \in Arc(i,j)} w_{i,j,e} \)

**Maximize:** \( \Sigma_i P(i).(\Delta(i) - TL(i,j, v_i)) \)

**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 \( e \in I(i) \), 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 \( \sigma \) 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 \( \sigma \), \( v_j T_j \) and values \( \Pi((j,e)) = \text{access level to arc } e \) that job \( j \) can transmit to \( i_0 \) are available. Then it applies a 3 stage function \( \text{Assign}(i_0) \) which computes (see Fig. 1) \( v_{i_0}, T_{i_0} \) and flow values \( w_{j,i_0,e} \), \( j \neq i_0 \), and \( e \in \text{Arc}(j,i_0) \), or, in case of failure, a job \( j \)-fail \( \sigma \) considered as cause of the failure.

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

**Assign1**

For \( e \) in \( \Pi(i_0) \) do

Let \( L-Job = \{j \; s.t \; (j \; \sigma \; i_0) \; \text{AND} \; (e \in \text{Arc}(j,i_0) \; \text{AND} \; (\Pi((j,e)) \neq 0)) \; \text{ordered according to increasing} \; T_j + TL((j,i_0),v) \} \); \( v \leftarrow 0 \); Not Stop ;

While \( L-Job \neq \text{Nil} \; \text{AND} \; \text{Not Stop} \)

If \( T_j + TL((j,i_0),v) + L_{TIME}(x_0) + P((i_0),(v+\Pi((j,e))) \leq \Delta(i_0) \); then

Compute \( w \) such that \( T_j + TL((j,i_0),v(j)) + L_{TIME}(x_0) + P((i_0),(v+\Pi((j,e)))) = \Delta(i_0) \); Stop ; \( v \leftarrow v + w \); \( w_{j,i_0,e} \leftarrow w \);

Else \( v \leftarrow -\Pi((j,e)) + v \); \( w_{j,i_0,e} \leftarrow -\Pi((j,e)) \); If Not Stop then Fail : Choose j-Fail in L-Job

Else \( v-\text{aux}(e) \leftarrow v \); If Not Fail then \( v_{i_0} \leftarrow \text{Sup } v-\text{aux}(e) \); \( e_0 \leftarrow \text{Arg Sup.} \)

\[
\sigma = x_1, x_2, ...; x_{i_0}, ...; x_{i_1}, ...; x_{i_2}, ...; x_{i_3}, ...; x_{i_4}, ...
\]

\( \Pi(x_0) = \{e_1, e_2\}; \text{CAP}(e_1) = 20, \text{CAP}(e_2) = 25; \Delta(x_0) = 50; L_{TIME}(x_0) = 10; \text{Arc}(x_2, x_0) = \{e_1\}; \text{Arc}(x_1, x_0) = \{e_1, e_2\}; \text{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; \]

**Figure 2:** Assign Process.

- (2): Assign1 computes \( v_{i_0} \) and, for any \( e \neq e_0 \) in \( \Pi(i_0) \)

a value \( v-\text{aux}(e) \) which may be less than \( v_{i_0} \); So Assign2 increases the \( w_{j,i_0,e} \) for \( e \neq e_0 \) in order to make job \( i_0 \) run at the same rate for all arcs \( e \) of \( \Pi(i_0) \). This part of the Assign process may induce a failure which Assign2 assign to some job \( j \)-Fail.

- (3): Assign3 makes the number of arcs provided with non null \( w_{j,i_0,e} \) values by shifting values \( w_{j,i_0,e} \) which involve, for a given \( j \), only one arc \( e \), to another job \( j' \) such that \( e \in \text{Arc}(j', i_0) \), \( w_{j',i_0,e} \neq 0 \) and \( \Pi(j'; e) \geq w_{j,i_0,e} \).

Then Greedy-NPEP comes as follows:

**Greedy-RCPSP-TL(\( \sigma \))**:

\( T_{j} \leftarrow 0 ; \text{For any } e \text{ do } \Pi(s,e) \leftarrow \text{CAP}(e) ; \text{Not Stop} ; \text{While (Not Stop) and \( \sigma \) no fully scanned do Apply Assign to current } i_0 \text{ and partial schedule; If (Success(Assign)) then } \text{For } e \text{ in } \Pi(i_0) \text{ and } (j \neq i_0) \text{ AND } (e \in \text{Arc}(j,i_0)) \text{ do } \Pi(i_0,e) \leftarrow v_{i_0}, \Pi(j,e) \leftarrow \Pi(j,e) - w_{j,i_0,e}; \text{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 \( \sigma \).

- **Initialization of \( \sigma \)**. For any \( i \), we set \( \text{SME}(i) = (\Delta(i) - L_{TIME}(i) - 2P(i)(v_{\text{max}}+v_{\text{min}}),i) \), and compute \( \sigma \) by randomly sorting \( N^+ \) in such a way that if \( P(i) < P(j) \) and \( \text{SME}(i) < \text{SME}(j) \), then \( i < j \).

- **Making \( \sigma \) evolve**. In case of failure, Greedy-NPEP returns a pair \( (j-Fail, i_0) \), and this pair is inserted into a Tabu like set \( \text{FORBID} \) whose meaning is: If \( (j, i) \) is \( \text{FORBID} \), then we should have \( i < j \).

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

**Procedure NPEP-First-Step(Max-Iter; Threshold)** Initialize \( \sigma \) as described above ; \text{FORBID} \leftarrow \text{Nil} ; \text{Iter} \leftarrow 0 ; \text{Not Stop} ; \text{Success} \leftarrow 0 ; \text{While (Iter} \leq \text{Iter-Max) AND (Not Success) do Generate } \sigma \text{ consistent with } \text{FORBID} \text{ and Apply Greedy-NPEP; If Failure then Search a failure responsible (j-Fail, } i_0) \text{ pair and put into } \text{FORBID}. \)
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 \(L_{TIME}(i) + P(i)\nu_i\) 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 \(\lambda = \lambda(i), i \in N^*_i \geq 0\), such that \(\sum_i P(i). T^*_i = \sum_i \lambda(i)\nu_i + \) Constant.; Vector \(Grad = (Grad_i = - \lambda(i)\nu_i^2, i \in N^*_i)\) is a sub-gradient vector;
- Then we modify \(v\) and \(w\) according to (I1): \(v \leftarrow v \circ V\); \(w \leftarrow w + W, \) with \(V\) and \(W\) s.t. \(V.Grad < 0\) and \(v \circ V\) and \(w + W\) comply with (E4, E5, E6) and computed by solving Project-Grad following linear program:

\[
\text{Project-Grad}(U-Active, v, w, \delta, \text{Grad}) \text{ LP}: \]

\[
\begin{align*}
\text{[Compute V=} (V_i, i \in N^*_i), \text{and W=} (W_{i,j,e}, (i, j) \in U-Active, e \in \text{Arc}(i, j)) \text{ such that; } \\
&\forall (i, j, e), w_{i,j,e} + W_{i,j,e} \geq 0; \\
&\forall i \neq j, e \in \text{IP}(i), \Sigma_j W_{i,j,e} = \Sigma_j W_{j,i,e} = V_i; \\
&\forall e, \Sigma_i W_{i,j,e} = \Sigma_i W_{j,i,e} = 0; \\
&\forall i \neq j, e \text{-Min}(i) \leq v_i + V_i \leq \text{v-Max}(i); \\
&2.\delta \geq \Sigma_{i,j,e} V_i \text{ Grad}(i) \geq \delta) \\
\end{align*}
\]

Then NPEP-Second-Step comes as follows:

\textbf{Procedure NPEP-Second-Step:}

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

\textbf{V. NUMERICAL EXPERIMENTS.}

\textbf{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.

\textbf{Instances/outputs:} An instance is a path collection \(\{I(i), i \in N^*_i\}\), given together with values \(P(i), \Delta(i)\) and \(TIME(e_i)\). It is summarized by a 3-uple: \((n, m, \alpha)\), where \(n = \text{Car}(N^*_i), m = \text{number of arc} e, \) 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.

\textbf{Outputs:} For every 10 instance package, we compute:

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

\textbf{Table below provides results for} \(\alpha \in [1,2].\)

\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Inst. 1:} & \textbf{n =} & \textbf{20}, \textbf{m =} & \textbf{10} & \textbf{CPU-} \\
\hline
\textbf{\(\alpha = 1.2\)} & \textbf{22.30} & \textbf{13.8} & \textbf{4.7} & \textbf{40.4} \\
\textbf{\(\alpha = 1.5\)} & \textbf{2.50} & \textbf{29.5} & \textbf{13.0} & \textbf{12.3} \\
\textbf{\(\alpha = 1.7\)} & \textbf{1.39} & \textbf{40.8} & \textbf{17.7} & \textbf{8.1} \\
\textbf{\(\alpha = 2.0\)} & \textbf{1.08} & \textbf{61.7} & \textbf{19.3} & \textbf{5.2} \\
\hline
\textbf{Inst. 1:} & \textbf{n =} & \textbf{30}, \textbf{m =} & \textbf{15} & \textbf{CPU-} \\
\hline
\textbf{\(\alpha = 1.2\)} & \textbf{40.6} & \textbf{14.6} & \textbf{5.6} & \textbf{70.5} \\
\textbf{\(\alpha = 1.5\)} & \textbf{6.60} & \textbf{30.2} & \textbf{14.5} & \textbf{19.5} \\
\textbf{\(\alpha = 1.7\)} & \textbf{2.05} & \textbf{42.3} & \textbf{19.1} & \textbf{12.0} \\
\textbf{\(\alpha = 2.0\)} & \textbf{1.19} & \textbf{65.5} & \textbf{22.5} & \textbf{7.9} \\
\hline
\end{tabular}

\textbf{Comment:} Tightening deadlines \(\Delta(i)\) improve solutions.

\textbf{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 \(I(i)\).

\textbf{REFERENCES}