Adapting the RCPSP framework to Evacuation Problems

Christian ARTIGUES[1], Emmanuel HEBRARD[2], Alain QUILLIOT[2,], Peter STUCKEY[3], Hélène TOUSSAINT[2]

[1] LAAS Laboratory, CNRS Toulouse, France
e-mail: artigues@laas.fr

[2]LIMOS laboratory, CNRS/UCAlermont-Ferrand, France
e-mail: alain.quilliot@isima.fr

[3]Monash University, Melbourne, Australia

Keywords: Scheduling, RCPSP, Evacuation.

1. Introduction

In the context of the H2020 GEOSAFE European project [7], we have been working on the *late evacuation problem*, that means the evacuation of people and eventually critical goods facing a natural disaster (flooding, wildfire..).

We did it accordingly to the 2-step approach currently favored by practitioners [2, 4, 7]: the first step (pre-process) computes the routes that evacuees will follow; the second step, to be performed in real time, schedules the evacuation of estimated late evacuees along those routes. In practice, performing this last step requires forecasting the evolution of the disaster, rather difficult in the case of wildfires, because of their dependence to topography and meteorology [4]. But we consider here this issue as resolved and focus on the priority rules and evacuation rates which have to be imposed to evacuees [3]. Our model *non preemptive Tree evacuation planning* problem (NPETP) is equivalent to the model proposed in [1] evacuees have been clustered into groups with same original location and pre-computed route, and once a group starts moving, it keeps on at the same rate until reaching his target safe area (*non preemption*. This last hypothesis derives from practical concerns and aims at avoiding any panic effect during the evacuation process. The pre-computed evacuation routes are supposed to define a tree, with evacuee groups located at the leaves of the tree and the safe target place at its anti-root. While [1] addresses the problem through a discretization of both time and rate domains and constraint programming techniques, we make it here appear as a RCPSP: *Resource Constrained Project Scheduling Problem* variant, [5,6]), and use this RCPSP reformulation in order to get accurate optimistic bounds (lower bounds) and design an efficient network flow based heuristic.

The paper comes as follows: Section 2 provides the NPETP model. Section 3, 4 are devoted to optimistic bounds and algorithms. Section 5 proposes numerical tests.

2. The RCPSP Oriented NPETP Model

We consider here a tree A, oriented from its leaves towards its anti-root (*safe* target node), which is the extremity of a single arc *Root.* The leaf set is denoted by $J = \{1..N\}$: every $j \in J$ is provided with an *evacuee population* $P(j)$ which has to be brought until the safe target anti-root. We indistinctly talk about j as an *evacuation node* and an *evacuation job*. Arcs e are provided with both a capacity $CAP(e)$ and a length (or duration) $L(e)$. This induces that the path $\Gamma(j)$ which connect j to the anti-root has a length $\Lambda(j)$. Values $CAP(e)$ increase as long as we advance along path $\Gamma(j)$. For any arc e, $J(e)$ denotes the subset of J defined by all j such that $e \in \Gamma(j)$.

With every population j is associated a *deadline* $\Delta(j)$: evacuation of j must be achieved no later than time $\Delta(j)$. The evacuation time of population j is determined by its speed, which is supposed to be the same for all evacuees, and by its evacuation rate (number of people/time unit) v_j, which is imposed to be independent on the time. It comes that the duration of *evacuation job j* is equal to $\Lambda(j) + P(j)/v_j$. We want to schedule the evacuation process, while meeting the following requirements:

- Every evacuation job j is achieved at time T^{End}_j no later than deadline $\Delta(j)$;
- For any arc e, the sum of the *evacuation rates* v_j, taken for all j which are concurrently entering on e, does not exceed $CAP(e)$; (E1)
- The global *safety margin* $M = \mathrm{Inf}_j\,(\Delta(j) - T^{End}_j)$ is the largest possible.

In order to cast our NPTEP problem into the RCPSP framework, we identify every *evacuation job j* with the *entering* process defined by the arrival of evacuees of j on the arc *Root*. Let us denote by T_j the starting time of this process and by T^*_j its ending time. Then we get a schedule if we decide, for every $j \in J$, starting time T_j, ending time T^*_j, and evacuation rate v_j, in such a way that:

- T_j is no smaller than the *release date* $R(j)$ = distance in the tree A from node j to the origin of *Root*;
- $T^*_j = T_j + P(j)/v_j \leq \Delta(j) - L(Root)$, which becomes the *deadline* $D(j)$ of job j. We deduce that v_j should be no smaller than $vmin(j) = P(j)/(D(j) - R(j))$.
- Above capacity constraints (E1) are never violated.

Because of the *Non Preemption* hypothesis, *entering* process j should take place in a continuous way between time T_j and time T^*_j, and define an interval. So we simplify the formulation of (E1) by introducing a vector $Z = (Z_{j,k},\ j,k = 1..N)\ 1..N)$ such that $Z_{j,k} = 1$ iff j *precedes k*. This allows to get our RCPSP oriented NPTEP model as follows:

NPTEP Model: {Compute vectors $T = (T_j\ ,\ j = 1..N)$, $T^* = (T^*_j\ ,\ j = 1..N)$, $v = (v_j\ ,\ j = 1..N) \geq 0$, and {0,1}-valued vector $Z = (Z_{j,k},\ j,k = 1..N)$ such that:

- $Z_{j,k} = 1$ iff *j precedes k*: we say that j and k *overlap* if $Z_{j,k} + Z_{k,j} = 0$;
- *Temporal constraints*:
  - For any j, $T_j + P(j)/v_j = T^*_j \leq D(j)$;
  - For any j, $T_j \geq R(j)$;
  - For any j, k, $Z_{j,k} = 1 \rightarrow T^*_j \leq T_k$;
- *Resource constraints*:
  - For any arc e, and any clique $C \subseteq \{1..N\}$ in the *overlap* sense,

    $\sum_{j \in J(e) \cap C} v_j \leq CAP(e)$. (E2)
- *Safety Margin Criterion*: Maximize $M = \mathrm{Inf}_j\,(D(j) - T^*_j)$}

3. Optimistic Upper Bounds

We propose 2 upper bounds, both derived from the relaxation of *the Non Preemption constraint from the NPTEP model. We get upper bound UB-Tree* while keeping all constraints but the *Non Preemption* Constraint; we get upper bound *UB-Arc* while also relaxing all constraints (E2) but those related to the final arc *Root* and those related to the arcs $e(j)$ whose origins are the leaves $j = 1..N$ and whose capacities $CAP(e(j))$ are upper bounds values for the *evacuation rates* v_j. Computing both *UB-Arc* and *UB-Tree* follows the same algorithmic scheme:

Start from time value $t = 0$;
At any time t, consider all (*entering*) jobs j which have not been achieved yet and which are such that $t \geq R(j)$; Denote by $Q(j)$ the population which remains to enter into the arc *Root*;
Compute, for any such a job j, its current optimistic *safety margins* M_j, which means the *safety margin* $D(j) - T^*j = D(j) - Q(j)/CAP(e(j))$ which would be achieved if constant rate $v_j = CAP(e(j))$ were applied to job j from t on;
Make run jobs j with higher value M_j, which are assigned values v_j in such a way that values M_j evolve at the same pace for those jobs with highest priority;

Compute smallest time value t^* which fits some of the 3 following situations: (a) some job j gets to its end; (b) t coincides with the release date $R(j)$ of a job j which could not be started before; (c) the priority order related to safety margins M_j has been modified.
Update t: t <- t^*.

4. Algorithms

We propose here 2 algorithms. The first one is a fast insertion algorithm which relies on the Network Flow approach which was implemented in [6] in the case of RCPSP. The second one was already described in [1] and involved the use of IBM *CP Optimizer* Software.

4.1. A Network Flow Oriented Heuristic *NPETP*.

The key idea here is to consider the arcs e of the tree A as resources, likely to be exchanged by *evacuation jobs* i, j whose paths $\Gamma(i)$ and $\Gamma(j)$ share arc e. According to this purpose, we extend above NPETP model by introducing, for any pair (i,j) and any arc e in the set $Arc(i,j) = \Gamma(i) \cap \Gamma(j)$, the part $w_{i,j,e}$ of access rate to e which is given by i to j. We see that resulting vector w has to comply with the following flow constraints (E3):

- For any $j = 1..N$, e in $\Gamma(i)$: $\Sigma_{\text{i such that } e \in \text{Arc(x,y)}}\ w_{i,j,e} = v_{\text{i}} = \Sigma_{\text{i such that } e \in Arc(j,i)}\ w_{j,i,e}$;   (E3)

We see that the main difficulty here is that we must choose between assigning high *rates* v_{j} to *jobs j* and let them monopolize the access to transit arcs of A, or conversely restricting v_{j} in order to make j share its arcs. In order to deal with it we design a 2 step *NPETP* approach:

***NPET*P Algorithmic scheme**:

First step (conservative approach):

Starts from deadlines $D(j)$, $j = 1..N$; Not *Stop*;

While Not *Stop* do

Look for a feasible Schedule (T, v, T^*);

If *Fail* then *Stop* Else decrease deadlines $D(j)$, $j = 1..N$, in order to force values T^*_j to decrease and so improve the *Safety Margin* criterion.

Second step: Improve the solution by making evacuation rates v_j increase (and so dates T^*_j decrease), through resolution a specific linear program on vectors w and v.

Then the core of *NPETP* Algorithm is related to the "Look for a feasible Schedule (T, v, T^*)" instruction of the "While" loop of the first step. We do it while relying on above flow vector w and providing every job with no more than what it needs in order to be achieved in time:

Start from some linear ordering σ defined on N; Not *Success*; Not *Failure*;

While Not *Success* and Not *Failure* do

Scan σ: j_0 being current job, values v_{j}, T_j and $\Pi(j,e)$ = access level to arc e that job j can transmit to j_0 have been computed for any j prior to j_0 in s;

Then:

(1) : Scan path $\Gamma(j_0)$: for any e in $\Gamma(j_0)$, compute flow values $w_{j,j0,e}$, j prior to j_0 in σ, in such a way $T^*_{j0} \le D(j_0)$; Derive $v_{j0} = \text{Sup}_{\text{e}} (\Sigma_j\, w_{j,j0,e})$ and related arc e_0;

(2) : Increase the $w_{j,j0,e}$ for $e \neq e_0$ in order to make *job* j_0 run at the same *rate* for all arcs e of $\Gamma(j_0)$.

If Not *Success* then modify σ accordingly and update *Failure*.

4.2. A Constraint Programming Approach for a Discrete Version of the NPTEP Model.

This approach associates with every variables v_j, T_j, $j = 1..N$, finite discrete domains, and apply the constraint propagation techniques which are at the core of the IBM *CP Optimizer* Software. All details are provided in [1]. Because of the rounding of values v_j, T_j, $j = 1..N$, it is also a heuristic approach.

5. Numerical Experiments

Purpose/Technical context: Algorithms were implemented C++, Windows 10, Visual Studio 2017, on PC with 16Go de RAM, Intel Core i5-8400 CPU @ 2.80GHz. Our goal was to evaluate both ability of the *NPETP* algorithm to yield good solutions and the accuracy of the optimistic upper bounds of Section III, while using results obtained in [1] through constraint programming (CPO Optimizer) as reference results.

Instances/outputs: They are as in [1]. The main characteristics of an instance is the number *N* of populations. We consider several instance packages, with, for any package, the number *S* which denotes the number of instances inside the package.

Outputs: For every instance package, we compute:

- *resCPO* = reference value through IBM-CPO in no more than 100 s (CPU).
- *optCPO* = number of instances such that IBM-CPO could achieve optimality of the discrete approximation of *NPETP*.
- *NPETP* = Value obtained through *NPETP* Algorithm; *cpuNPEP* = Related CPU time.
- *UB1* = Optimistic (upper) bound *UB-Arc*; *UB2* = Optimistic bound *UB-Tree*.

CPU times for the computation of both *UB1* and *UB2*, since they never exceed 0.1 s.

Then the following table provides a summary of our results:

N	*S*	*resCPO*	*optCPO*	*NPETP*	*cpuNPETP(s)*	*UB1*	*UB2*
10	15	104,00	12,00	97,96	0,56	112,16	107,16
15	16	69,81	12,00	65,14	0,79	78,53	73,94
20	11	40,09	8,00	42,36	1,30	51,28	43,58
25	5	8,40	0,00	43,80	1,17	70,30	49,20

Comments: The model handled by IBM-CPO is an approximation of NPETP model, and so *NPETP* algorithm obtains in some cases better results that IBM-CPO, even when IBM-CPO concludes to optimality. In any case, *NPETP*, whose computation times are very small, outperforms IBM-CPO as soon as the size of the problem increases. We also see that the *Tree Upper Bound UB2* provides us with a very efficient estimation of the optimal *NPETP* value, since the gap between *UB2* and Inf(*resCPO, NPETP*) is in average 5% (it tends to increase with the size of the instance).

Acknowledgements

This work is partially funded by the H2020-MSCA -2015 U.E project GEO-SAFE.

References

[1]. Artigues.C, Hebrard.E, Pencolé.Y, Schutt.A, Stuckey.P: “A study of evacuation planning for wildfires”; 17 th Int. Workshop on Constraint Modelling/Reformulation, Lille, France, (2018).
[2]. Bayram.V : “Optimization models for large scale network evacuation planning and management : a review “; Surveys in O.R and Management, (2016)
[3]. Even.C, Pillac.V, Van Hentenryk.P: “Convergent plans for large scale evacuation”; In Proc. 29 th AAAI Conf. On Artificial Intelligence, Austin, Texas, p 1121-1127, (2015).
[4].Intini.P, Gwynne.S.M, Ronchi.E, Pel.A : “Traffic modeling for wildland urban interface fire evacuation” ; Jour. Transportation Eng. A, 145, 3 (2019)
[5].Orji.M.J, Wei.S. “Project Scheduling Under Resource Constraints: A Recent Survey”. Inter. Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 2, (2013)
[6]. Quilliot.A, Toussaint.H: “Flow Polyedra and RCPSP”, RAIRO-RO, 46-64, p 379-409, (2012)
[7]. Veeraswamy.A, Galea.E, .Filippidis.L, Lawrence.P, Haasanen.S, Gazzard.R.J, TSmith.T.E: “The simulation of urban scale scenarios with application to the Swinly forest fire”; Safety Science 102, p 178-193, (2018).