Heuristics for Scheduling Evacuation Operations in Case of Natural Disaster

Kaouthar Deghdak, Vincent T'kindt and Jean-Louis Bouquard
Université François–Rabelais Tours, Laboratoire d'Informatique (EA 6300),
Equipe Ordonnancement et Conduite (ERL CNRS 6305), 64 Avenue Jean Portalis, 37200 Tours, France

Keywords: Evacuation, Greedy Heuristic, Matheuristic.

Abstract: In this paper, we consider a large-scale evacuation problem after an important disaster. The evacuation is assumed to be done by means of a fleet of buses, thus leading to schedule the evacuation operations by buses (Bus Evacuation Problem, BEP). We propose time indexed formulations, as well as heuristic algorithms like greedy algorithms and a matheuristic. This matheuristic uses the former formulation to improve the best solution obtained by the greedy heuristics. In computational experiments, we analyse and evaluate the efficiency of the proposed solution algorihms.

1 INTRODUCTION

After an important disaster, the evacuation of people from the damaged area to a safety area becomes necessary. From the literature, it turns out that two main application scenarios have been considered: the scenario of building evacuation and the scenario of region evacuation. Evacuation of buildings is extensively discussed in (Chalmet et al., 1982) and (Hamacher and Tjandra, 2001). Several reviews tackle the evacuation problem for large regions, e.g, (Sattayhatewa and Ran, 2000), (Church and Sexton, 2002), (Sbayti and Mahmassani, 2006) and (Bish et al., 2013).

The evacuation models proposed in the literature can be classified into two categories: simulation-based models and optimization-based models. Simulation-based models are developed to define, analyse and evaluate evacuation plans, e.g, (Sheffi et al., 1982) and (Kwon and Pitt., 2005). These models are generaly based on the Dynamic Traffic Assignment (DTA) methodology, e.g, (Iliaskopoulos., 2000), (Mahmassani, 2001) and (Sbayti and Mahmassani, 2006). Optimization-based models are mainly based on dynamic network flow models, like universal maximum flow, minimum dynamic cost flow, maximum dynamic flow, quickest flow and quickest path, e.g, (Yamada, 1996) and (Hamacher and Tjandra, 2001). The goal of the evacuation problems is to find the optimal evacuation plan for an emergency situation. A proper evacuation plan is imperative to reduce fatalities. To solve these problems, numerous optimization models have been proposed, in which we find three objective functions:

1. Minimizing the Total Evacuation Time (TET): TET is the period during which the evacuees will be exposed to risk until reaching a safe destination.
2. Minimizing the Clearance Time (CT): CT is the time until the last vehicle or evacuee leaves the damaged area.
3. Maximizing the number of people exiting the damaged area in a given time period.

The first objective function is equivalent to the second. They are both mentionned because they are presented in differenent ways in the literature. In some cases, the third objective function becomes more important than the first two. For instance, when the total evacuation time or the clearance time are fixed in advance by the authorities.

(Bish, 2011) introduced and studied a new model for bus-based evacuation planning. The choice of buses as a transportation mean has been motivated by the fact that bus-based evacuation is logistically complex, expensive, produced unacceptable levels of congestion and more dangerous than the bus-based evacuation should be.

The work dealt with in this paper follows the same line. We consider the evacuation of people due to a naturel disaster like earthquakes, where evacuees have to change their centre of lives from several days

Deghdak K., T'kindt V. and Bouquard J..
Heuristics for Scheduling Evacuation Operations in Case of Natural Disaster.
DOI: 10.5220/0004922102940300
In *Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems* (ICORES-2014), pages 294-300
ISBN: 978-989-758-017-8

to several months with the eventual goal of returning back to their respective home. In particular, we assume that the locations of gathering points (i.e where people are evacuated, outside the damaged area), the locations of collection points (i.e where people are gathered waiting to be evacuated) and the capacities of the transportation network are known. The goal is to define a macroscopic plan of evacuation which means that people are considered homogeneously, i.e. the evacuees are assumed have the same behaviour and have to be transported from the collection points to the gathering points in a minimal amount of time. The evacuation is done by means of a set of homogeneous buses.

The remainder is organized as follows. In Section 2, we describe the Bus Evacuation Problem in details and provide a mixed-integer programming formulation. In Section 3, we present greedy heuristics for the problem, and an iterative local search heuristic, referred to as matheuristic, in Section 4. Computational results are presented in Section 5. Finally, Section 6 concludes the paper.

2 PROBLEM DEFINITION AND MATHEMATICAL FORMULATION

In this section, we first describe the Bus Evacuation Problem (BEP) and next provide a time-indexed formulation. This formulation is used in the following to evalute the quality of the proposed heuristics and to develop a matheuristic algorithm.

2.1 Problem Definition

Consider a network ($\mathcal{N}$, A), where $\mathcal{N}$ and A respectively denote the set of nodes and edges. $\mathcal{N}$ is composed of two subsets of nodes: *CP* and *GP*. *CP* is a set of collection points where evacuees are initially located and *GP* is a set of gathering points. An edge $(i, j) \in$ A exists iff evacuees can be transported from collection point i to gathering point j. Each collection point i has a demand e_i $i \in CP$, which is the number of buses required to evacuate all people at point i. Each gathering point j has a capacity cap_j, $j \in GP$, wich is the number of evacuees that can be brought to j: it is expressed as a number of buses. Each edge (i, j) has a non-negative length $p_{i,j,t}$, which is the travel time between each couple (CP_i,GP_j) and it is time-dependent. This means that it can be increased or decreased, over time, depending on the state of the network. This is a consequence of the evolution of the transportation network through time due to events like earthquake replicates, road repairs, roads' congestion.... We assume that events that modify the transportation network, may occur: we consider that such k events happen at a known time d_l and can change the value of travel times between a collection point i towards gathering point j:

$$p_{i,j,t} = \begin{cases} a^0_{i,j} & \text{if } t_{ij} \in]0, d_1] \\ a^1_{i,j} & \text{if } t_{ij} \in]d_1, d_2] \\ \vdots & \\ a^{k-1}_{i,j} & \text{if } t_{ij} \in]d_{k-1}, d_k] \end{cases}$$

where $a^x_{i,j}$ is the travel time if the evacuees are transported from a collection point i to gathering point j in the x^{th} time interval, i.e. it's starting time $t_{ij} \in]d_{x-1}, d_x]$. The number of finite intervals $[d_{l-1}, d_l]$ is determined by a preliminary forecasting of the evolution of the transportation network.

A fleet of K identical buses is given and used to evacuate people. The problem we consider is to find a schedule such that all evacuees are transported from the collection points *CP* to the gathering points *GP*, minimizing the total evacuation time denoted by C_{max}.

The Figure 1 depicts an example of a bus evacuation network at time t, with three collection points, and two gathering points. The number of buses required for evacuees in the collection points are 2, 4 and 5, while the gathering points' capacities are 6 and 9 buses of evacuees.

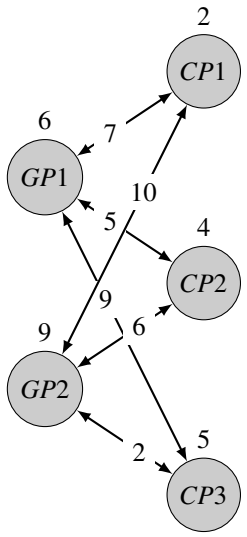

Figure 1: Exemple of BEP network at a given time t.

2.2 A Time-indexed Formulation

The Bus Evacuation Problem can be modeled as a cumulative scheduling problem with additional resource

constraints: the gathering points are cumulative parallel machines and the jobs to schedule are people to evacuate. The additional ressources are the buses which are disjunctive resources. As mentioned earlier in Section 2.1, an important feature of our evacuation scheduling problem is that the processing times (i.e.travelling times) of the jobs are dependent on the starting times of the job in the schedule.

Consider that we have M parallel and cumulative machines, each machine j corresponding to a gathering location GP_j, with a capacity cap_j expressed as a number of buses that can bring evacuees. $N = \sum_{i \in CP} e_i$ evacuation operations have to be scheduled on the machines, and each operation i is associated to a collection point CP_ℓ. Similarly to the calculation of the machine's capacities, it is assumed that one evacuation operation coressponds to the evacuation of people by a full bus. We suppose that the capacity of the reconfigured network is sufficient after the disaster to enable a fluent evacuation. This hypothesis enables to approximate the bus routting by considering only travel times.

To model the Bus Evacuation Problem we have proposed a time-indexed mathematical formulation. Usually, time-indexed formulations on scheduling problems yield simple and efficient models despite the presence of a pseudo-polynomial number of variables (see (Berghman et al., 2011) among others). Throughout the paper, we make use of the notation $[S] := \{1, \ldots, S\}$. Let us turn to the model for our evacuation scheduling problem in which T is the time horizon. The decision variables are:

$$\forall i \in [N], \forall j \in [M], \forall t \in [T];$$

$$x_{i,j,t} = \begin{cases} 1 & \text{if a bus starts the evacuation operation} \\ & i \text{ towards } j \text{ at } [t, t+1[. \\ 0 & \text{otherwise.} \end{cases}$$

and C_{max}, the duration of the schedule.

The proposed (IP) formulation is as follows:

$$\min C_{max}. \quad (1)$$

Subject to:

$$C_{max} \geq (t + p_{i,j,t}) x_{i,j,t}$$

$$\forall i \in [N], \forall j \in [M], \forall t \in [T] \quad (2)$$

$$\sum_{t \in [T]} \sum_{i \in [N]} x_{i,j,t} \leq cap_j \qquad \forall j \in [M] \quad (3)$$

$$\sum_{i \in [N]} \sum_{j \in [M]} \sum_{\substack{t' \in [0,t] \\ p_{i,j,t'} + t' > t}} \quad (4)$$

$$x_{i,j,t'} \leq K \qquad \forall t \in [T] \quad (5)$$

$$\sum_{t \in [T]} \sum_{j \in [M]} x_{i,j,t} = 1 \qquad \forall i \in [N] \quad (6)$$

$$x_{i,j,t} \in \{0,1\} \quad (7)$$

$$C_{\max} \in \mathbb{N} \quad (8)$$

Constraints (2) define the value of criterion C_{max}, which is then minimized by the objective function (1). Constraints (3) are the gathering point capacity constraints: we cannot exceed the capacities of gathering points. Constraints (5) are the bus capacity constraints : we cannot exceed the number of buses we have. Constraints (6) ensure that each operation is processed once and only once. Constraints (7) and (8) are the logical binary and non-negative integer restrictions on the $x_{i,j,t}$ and $C_{\max}$ variables, respectively.

Notice that when the processing times are not time-dependant and the capacities of the gathering points are sufficiently large enough, then the BEP reduces to the identical parallel machine scheduling problem with makespan minimization. The latter being strongly $\mathcal{NP}$-hard (see (Garey and Johnson, 1979)), we can deduce that BEP is so.

3 GREEDY HEURISTICS

In this section, we describe several greedy heuristic algorithms to solve the Bus Evacuation Problem. These heuristics aim to compute a solution in a very fast way. We outline the need for such heuristic procedures to solve real-size instances which are very large.

Figure 2 presents the general algorithm of the proposed greedy heuristic, which takes upon entry a sequencing rule $\mathcal{R}$. Let *oprlist* be the set of the operations that will be assigned on machines, *sortlist(i,j)* be the set of sorted operations according to the rule $\mathcal{R}$ at a given time t, and C^m_{max} be the makespan of a machine m. The greedy heuristic's solution is stored in *schedulelist*. Each operation added in *schedulelist* will be deleted from *oprlist* and *sortlist(i,j)*.

The function event, presented in the pseudo code of the greedy algorithm (Figure 2), checks if there is an event requiring to recalculate the priority list *sortlist*. Each rule $\mathcal{R}$ has a specific event function.

In the following, we have tested eight greedy heuristic versions.

1. Version 1: this version uses the rule $\mathcal{R}_1$. To each operation opr_i, let tno_i be the total number of operations of the same job j than opr_i. Then, the operations are sorted by non-decreasing order of their value tno_i. The function event always returns false because the tno_i of each operation opr_i does not change throughout the algorithm.

Generic Greedy Algorithm ($\mathcal{R}$)

Input: /* A sorting rule $\mathcal{R}$*/
$schedulelist \leftarrow \emptyset$, $sortlist \leftarrow \emptyset$,
$oprlist \leftarrow \{opr_1, opr_2, ..., opr_N\}$;
$t \leftarrow 0$, $k \leftarrow |K|$.
Arrange operations in the *sortlist(i,j)* using the rule $\mathcal{R}$.
Repeat
  Add the k first operations in *schedulelist* (i,j,t);
  Delete k operations added in *schedulelist* from *oprlist* and *sortlist* ;
  if t=0 **then** $k \leftarrow 0$;
  Let $m \in [M]$ be a machine with the minimum C_{max};
  $t \leftarrow C^m_{max}$;
  $k \leftarrow k+1$;
  if event() **then**
    $sortlist \leftarrow \emptyset$;
    Arrange operations in the *sortlist(i,j)* using the rule $\mathcal{R}$;
  end if
Until $|oprlist| = 0$ and $t > T$
  return Feasible solution *schedulelist* .

Figure 2: Greedy-Heuristic algorithm.

2. Version 2: this version uses the rule $\mathcal{R}_2$. In this rule, operations opr_i are sorted by decreasing order of their value tno_i. The function event always returns false because the tno_i of each operation opr_i does not change throughout the algorithm.
3. Version 3: this version uses the rule $\mathcal{R}_3$. The operations are sorted according to the shortest processing time first rule (SPT) known in scheduling theory (Smith, 1956). This rule has been adapted to take the time-dependency of the processing time $p_{i,j,t}$ into acount. The function event returns true if there are modifications in the values of operations' processing time.
4. Version 4: this version uses the rule $\mathcal{R}_4$. The operations are sorted in non-decreasing order of their ratio $\frac{nos_j}{tno_j}$, with nos_j the number of operations of a job j already scheduled and tno_j the total number of operations of a job j. If several operations have the same priority, we break ties by means of the SPT rule. The function event returns true if there is, at least, an operation opr_k that has a smaller ratio than the first operation in *sortlist*.
5. Version 5: this version uses the rule $\mathcal{R}_5$. In this rule, the operations are sorted in non-decreasing order of their ratio $\frac{tno_j}{nto} - \frac{nos_j}{tno_j}$, with nto is the total number of operations. The function event returns true if there is, at least, an operation opr_k that has a smaller ratio than the first operation in *sortlist*.
6. Version 6: this version uses the rule $\mathcal{R}_6$. To each operation opr_i, let nuo_i be the number of currently unscheduled operations of the same job j than opr_i. Then, the operations are sorted by decreasing order of their value nuo_i. The function event returns true if there is an operation opr_k that has a larger number of currently unscheduled operations than the first operation in *sortlist*.
7. Version 7: this version uses the rule $\mathcal{R}_7$. To each operation opr_i, let nuo_i be the number of currently unscheduled operations of the same job j than opr_i. Then, the operations are sorted in non-decreasing order of their value nuo_i. The function event returns true if there is an operation opr_k that has a smaller number of currently unscheduled operations than the first operation in *sortlist*.
8. Version 8: this version uses the rule $\mathcal{R}_8$. In this rule, we randomly assign a priority for each job. The function event returns true if there are modifications in the values of jobs' priority.

4 A MATHEURISTIC

In this section, we propose a local search method called matheuristic. The general idea of matheuristics is to exploit the strength of both metaheuristic algorithms and exact methods as well, leading to a hybrid approach (Della Croce et al., 2011). It is a heuristic based on mathematical programming.

We use the time-indexed formulation introduced in Section 2.2 to construct a matheuristic for BEP. Let be an initial heuristic solution given as the best solution obtained by the greedy heuristics. The matheuristic tries to improve that solution by exploring its neighborhood as follows. Let be a feasible schedule (heuristic solution) $\bar{x} =< \bar{x}_{i,j,t}, i \in [N], j \in [M], t \in [T] >$, where $\bar{x}_{i,j,t} = 1$, if operation i is processed on machine j at time t. We define a neighborhood $\mathcal{N}(\bar{x}, r, h)$ by choosing a date r in the schedule and a size parameter h. Let $\tilde{S}(r,h)$ be the index set of the operations scheduled in the time interval $[r, r+h[$. We call such a subset of operations an "operation-window".

The best solution in the neighberhood $\mathcal{N}(\bar{x}, r, h)$ is computed by minimizing the makespan C^w_{max}, subject to (3)-(8) and by adding the following constraint:

$$x_{i,j,t} = \bar{x}_{i,j,t} \quad \forall i \notin \tilde{S}(r,h), \forall j \in [M], t \in [0, r[\quad (9)$$

We call this reduced minimization problem the window reoptimization problem, and it is solved by a mathematical solver like CPLEX. The additional constraints (9) forces the changes to occur within the

operation-window. If we have an improvement in C^w_{max}, then, in the new solution $\tilde{x}$, all the operations which started after the time $r+h$ in the initial solution $\bar{x}$ will be left time shifted keeping their previous positions and respecting the model constraints.

If no improved solution is found a new operation-window (i.e. new value of r) is selected to be optimized until all possible windows have been selected. The search is stopped if no window reoptimization problem has an optimal solution which improves the current solution or if a predefined time limit is exceeded.

The algorithm of the matheuristic is given in Figure 3:

The Matheuristic Algorithm

Input:
$\bar{x}$ =<heuristic solution calculated by greedy algorithm>.
vect =<vector of starting times of all operations in $\bar{x}$ >.
Repeat
Set *improved* $\leftarrow$ *false*;
Set $i \leftarrow 0$;
Repeat
$r \leftarrow vect[i]$;
$\tilde{x} \leftarrow$ Neighborhood($\bar{x}, r, h$)
if ($C_{max}(\bar{x}) > C_{max}(\tilde{x})$) **then**
$\bar{x} \leftarrow \tilde{x}$
improved $\leftarrow$ *true*
For $r \leftarrow r+h$ **to** $C_{max} - r + h$ **do**
$\bar{y} \leftarrow$ Neighborhood($\bar{x}, r, h$)
if ($C_{max}(\bar{x}) > C_{max}(\bar{y})$) **then**
$\bar{x} \leftarrow \bar{y}$
end if
end For
recompute *vect* from $\bar{x}$
end if
Until $i \geq |vect|$ **or** improved **or** time limit expired
Until not improved **or** time limit expired
Return Feasible solution $\bar{x}$.

The Neighborhood Procedure

Calculate $\tilde{S}(r,h)$;
Reschedule operations in tha time window $[r, r+h[$ by minimizing $C^w_{max}(\bar{x})$ subject to (3)-(8), and (9);
Let $\tilde{y}$ be the solution obtained;
Return solution $\tilde{y}$.

Figure 3: Matheuristic algorithm.

5 COMPUTATIONAL EXPERIMENTS

In this section, we focus on the experimental evaluations of the (IP) solution, and the heuristic algorithms. We first describe how the experimentation have been setup.

Environment. All experiments were run on a computer with a 8-core Intel processor, running at 2.60 GHz with 20MB cache, 8 GB RAM and Windows 7. We wrote our code in C++ , and used the commercial IP solver CPLEX v.12.2. CPLEX was pinned to one core for the solution of the time-indexed formulation.

Dataset. This work is part research project, called DSS-Evac-Logistic, partially granted by the french research agency ANR. In this project, we consider the real-world instance of Nice (France), as a case study. So, the data sets are randomly generated in such a way that we always have a feasible and realistic instances for Nice city. The number of machines takes values $M \in \{2, 4, 6, 8, 10\}$ and the capacity cap_j of each machine draws at random from $\{20, 21, 22, ..., 40\}$. The number of jobs takes values $L \in \{10, 20, 30, 40\}$ and the number e_l of operations of job l are generated randomly from $\left[\frac{1}{4}\frac{0.9\sum_j cap_j}{L}; \frac{7}{4}\frac{0.9\sum_j cap_j}{L}\right]$. This generation ensures that the total operations' number is less than the total machines' capacity. We assume that the time period of evacuation is [D, D+1[, when D is the day of disaster. In the following, we take the time discretisation as a quarter of an hour, which implies that T is equal to 192 quarters.

We assume that we have 6 events happening and changing the value of the operations' processing time. To do so, we generated two degradation dates: $T^D_1, T^D_2 \in [0, 192[$, common to all operations, and three improvement dates: $T^A_1 \in U[0, T^D_1]$,$T^A_2 \in U]T^D_1, T^D_2]$, $T^A_3 \in U]T^D_2, 192[$, specific to each operation. Operations' processing time are drawn randomly from $\{2, 3, 4, ..., 8\}$. Finally, the number of buses $K = \frac{\sum e_l * \bar{p}}{180}$, where $\bar{p}$ is the average processing time, which ensures that we have enough buses to do the evacuation in 180 quarters of an hour.

For each instance, we run the following algorithms:

- Exact Solution Algorithm: we have tested the solution of the (IP) model by Cplex solver. In the remainder, this solution algorithm will be denoted by IP. Notice that in this model we use the best C_{max} value found by the heuristics, to set the value of the time horizon T. Besides, a time limit of 1800 seconds and a memory limit of 1 GB have been given to CPLEX solver.
- Heuristic Algorithms:
  - We have tested the eight versions of the greedy heuristic, and for each data set we keep the best solution found.
  - We have tested the matheuristic with a time limit of 600 secondes. Previous preliminary experimentations have shown that the best results

are obtained with the window size $h = 25$. The initial solution is the best solution found by the greedy heuristics

– We solve the time-indexed formulation with a time limit equal to that of the matheuristic. This heuristic is referred to as IPheuristic.

Table 1 presents the runnig times and the number of nodes explored by CPLEX when solving the IP. Column #oprs presents the number of operations, column #mach presents the number of machines, column #bus presents the number of buses, column #inst_tot presents the number of instances for each problem size, column #inst_opt presents the number of instances that have been solved to optimality by IP. Notice that the instances are randomly generated based on a number of jobs, and a number of machines. However, the size of the instances solved by IP depends on the total number of operations, i.e. $\sum_l e_l$. consequently, after having generated instances we have gathered them in classes of "equivalent size instances", but in term of number of operations, number of machines and number of buses. Columns IP (time) report the minimum, the average and the maximum running times of CPLEX when solving IP. IP (nodes) columns report the minimum, the average and the maximum explored nodes. As the results in Table 1 illustrate, for almost all the instances with less than 60 operations the IP find the optimal solutions. Unfortunately, when the number of operations varing between 90 and 110, IP can not solve the instances to optimality.

Table 1: Running times and number of explored nodes.

#oprs	#mach	#bus	#inst_tot	#inst_opt	IP (time(s))			IP (nodes)		
					min	avg	max	min	avg	max
[20,35[	2	[1,2]	32	32	15	23,72	87	0	217,7	901
[35,40[	2	2	17	16	14	63,06	200	0	243,5	784
[40,60[	2	[2,3]	31	30	17	106,8	384	0	367,7	4251
[40,60[	4	[2,3]	6	5	85	323	859	0	207,8	602
[60,70[	4	3	13	6	175	921,2	1710	0	359,5	872
[70,80[	4	[3,4]	17	1	1220	1220	1220	1015	1015	1015
[80,90[	4	4	26	1	610	610	610	147	147	147
[90,110[	4	[4,5]	18	0	/	/	/	/	/	/

In the following, we investigate the quality of the proposed heuristics.

Table 2 displays the results of IPheuristic compared to the best heuristic solutions delivred by IPheuristic or matheuristic solutions (the matheuristic is at least as good than the greedy heuristic) and its running time. Let dev(A) be the deviation associated to a heuristic A. We have:

$$dev(A) = \frac{C_{max}(A) - \min(C_{max}(A), C_{max}(bestheuristic))}{\min(C_{max}(A), C_{max}(bestheuristic))} * 100$$

IPheuristic performs well when the number of operations is less than 40 and requires less than 600 secondes to provide the best solutions. For the instances, for which the number of operations belongs to [40, 70[we observe a deterioration of the quality of this heuristic. For instance, in some cases, the IPheuristic solution is far from the best heuristic solution with a deviation of 95%. Furthermore, we found that for larger instances, IPheuristic becomes less competitive, and fails to produce good solutions whitin a time-limit of 600 secondes.

Table 2: Evaluation of the IPheuristic.

#oprs	#mach	#bus	dev%			CPU time (s)		
			min	avg	max	min	avg	max
[20,35[	2	[1,2]	0	0	0	17	49	101
[35,40[	2	2	0	0	0	98	217,8	600
[40,60[	2	[2,3]	0	0	0	53	277,8	600
[40,60[	4	[2,3]	0	1.35	42,06	575	597	600
[60,70[	4	3	0	1	3,84	600	600	600
[70,80[	4	[3,4]	38,64	71,24	115,25	600	600	600
[80,90[	4	4	26,42	64,37	105,1	600	600	600
[90,110[	4	[4,5]	23,77	58,73	95.35	600	600	600

The results presented in Table 3 show the quality of the solutions obtained by the greedy heuristics against the best heuristics solutions. All the proposed priority rules are tested on the small and large instances, and we have found that the greedy heuristics which use the rules $\mathcal{R}_3$ and $\mathcal{R}_4$ outperform the other rules. We note that running these heuristics takes a negligible amount of time.

Table 3: Evaluation of the greedy heuristics.

#oprs	#mach	#bus	dev %		
			min	avg	max
[20,35[	2	[1,2]	5,26	12,89	27,42
[35,40[	2	2	7,5	14,58	25,29
[40,60[	2	[2,3]	9	15,85	30,84
[40,60[	4	[2,3]	4,34	8,91	13,04
[60,70[	4	3	0	2,45	12,35
[70,80[	4	[3,4]	0	0,72	3,15
[80,90[	4	4	0	0,92	6,94
[90,110[	4	[4,5]	0	1,57	9,57

Table 4 provides the deviation of the matheuristic against the best heuristic solution. For instances with less than 70 operations in size, the matheuristic succeeds in providing the best solution on some instances since the minimal deviation value is 0. For instances with more than 60 operations in size, the matheuristic outperforms, on the average, the IPheuristic. For the instances with a number of operations in [60, 70[, the matheuristic provides the best solution for 12 out of 13 instances. When the number of operations belongs to [70, 80[, the best solutions are always delivred by the matheuristic within a time limit of 600 secondes.

To summarize all the results introduced so far, we can conclude that the matheuristic is the best heuristic algorithm, even if for large instances the time limit must be increased to enable improvement over the greedy heuristics. Noteworthy, these ones seem to be a good compromise between quality and CPU time, especially for larger instances.

Table 4: Evaluation of the matheuristic.

#oprs	#mach	#bus	dev %			CPU time (s)		
			min	avg	max	min	avg	max
[20,35[	2	[1,2]	0	5,2	10,29	32	157,8	370
[35,40[	2	2	0	8.6	24,14	97	326,2	603
[40,60[	2	[2,3]	0	7,97	20	166	399,7	600
[40,60[	4	[2,3]	0	3.57	11,11	600	600	600
[60,70[	4	3	0	0,38	5,05	600	600	600
[70,80[	4	[3,4]	0	0	0	600	600	600
[80,90[	4	4	0	0	0	600	600	600
[90,110[	4	[4,5]	0	0	0	600	600	600

6 CONCLUSIONS

In this work, we have studied the problem of scheduling evacuation operations that we have called the bus evacuation problem (BEP). We have provided a time-indexed formulation for this problem. Next, we have provided three heuristics, the first one is a set of greedy heuristics. The second one is a local search called matheuristic based on the mathematical formulation provided, which improves the best geedy heuristic solutions. The third one is provided by CPLEX within an imposed time limit. During the conference, we will present all the computationnal results for small and larger instances, i.e. instances with 6, 8, 10, 20, 30 machines and more than 110 operations.

ACKNOWLEDGEMENTS

This research has been supported by ANR-11-SECU-002-01, project DSS_EVAC_LOGISTIQUE (CSOSG 2011).

REFERENCES

Berghman, L., Leus, R., and Spieksma (2011). Optimal solutions for a dock assignment problem with trailer transportation. *Annals of Operations Research*.

Bish, D. (2011). Planning for a bus-based evacuation. *OR Spectrum*, 33(3):629–654.

Bish, D. R., Sherali, H. D., and Hobeika, A. G. (2013). Optimal evacuation planning using staging and routing. *J Oper Res Soc*, pages –.

Chalmet, L., Francis, R., and Saunders, P. (1982). Network models for building evacuation. *Fire Technology*, 18(1):90–113.

Church, R. and Sexton, R. (2002). *Modeling Small Area Evacuation: Can Existing Transportation Infrastructure Impede Public Safety?* University of California Santa Barbara, National Center for Geographic Information and Analysis, Vehicle Intelligence and Transportation Analysis Laboratory.

Della Croce, F., Grosso, A., and Salassa, F. (2011). A matheuristic approach for the total completion time two-machines permutation flow shop problem. 6622:38–47.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co, New York, NY, USA.

Hamacher, H. W. and Tjandra, S. A. (2001). Mathematical modeling of evacuation problems: A state of the art. *In Pedestrian and Evacuation Dynamics (Schreckinberg, M. and Sharma, S. D. eds)*, 1964:227–266.

Kwon and Pitt. (2005). Evaluation of emergency evacuation strategies for downtown event traffic using a dynamic network model. *Journal of the Transportation Research Board*, 1922:149155.

Mahmassani, H. S. (2001). Dynamic network traffic assignment and simulation methodology in advanced system management applications. *Networks and Spatial Economics*, 1:67.

Sattayhatewa, P. and Ran, B. (2000). *Developing A Dynamic Traffic Management Model For Nuclear Power Plant Evacuation*. Transportation Research Board, 79th Annual Meeting.

Sbayti, H. and Mahmassani, H. S. (2006). Optimal scheduling of evacuation operations. *Journal of the Transportation Reasearch Board*, 1964:238–246.

Sheffi, Y., Mahmassani, H., and Powell., W. B. (1982). A transportation network evacuation model. *Transportation Research Part A*, 16(3):209–218.

Smith, W. (1956). Various optimizers for single-stage production. 3(1-2):59–66.

Yamada, T. (1996). A network flow approach to a city emergency evacuation planning. *International Journal of Systems Science*, 27(10):931–936.

Ziliaskopoulos., A. K. (2000). A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation Science*, 34:37.