A Constraint-Based Model for Fast Post-Disaster Emergency Vehicle Routing

Roberto Amadini [1], Imane Sefrioui [2], Jacopo Mauro [1], Maurizio Gabbrielli [1]
[1] *Department of Computer Science and Engineering/Lab. Focus INRIA. University of Bologna, Italy*
[2] *Computer Science, Operational Research and Applied Statistics Lab. Faculty of Sciences, Abdelmalek Essaadi University, Tetuan, Morocco*

***Abstract* — Disasters like terrorist attacks, earthquakes, hurricanes, and volcano eruptions are usually unpredictable events that affect a high number of people. We propose an approach that could be used as a decision support tool for a post-disaster response that allows the assignment of victims to hospitals and organizes their transportation via emergency vehicles. By exploiting the synergy between Mixed Integer Programming and Constraint Programming techniques, we are able to compute the routing of the vehicles so as to rescue much more victims than both heuristic based and complete approaches in a very reasonable time.**

***Keywords* — Disaster Recovery, Decision Support Systems, Constraint Programming, Mixed Integer Programming.**

I. Introduction

Disasters are unpredictable events that demand dynamic, real-time, effective and cost efficient solutions in order to protect populations and infrastructures, mitigate the human and property loss, prevent or anticipate hazards and rapidly recover after a catastrophe. Terrorist attacks, earthquakes, hurricanes, volcano eruptions *etc.* usually affects a high number of persons and involve a large part of the infrastructures thus causing problems for the rescue operations which are often computationally intractable. Indeed, these problems have been tackled by using a *pletora* of different approaches and techniques, ranging from operational research to artificial intelligence and system management (for a survey please see [1]).

Emergency response efforts [2] consist of two stages: pre-event responses that include predicting and analyzing potential dangers and developing necessary action plans for mitigation; post-event response that starts while the disaster is still in progress. At this stage the challenge is locating, allocating, coordinating, and managing available resources.

In this paper we are concerned with post-event response. We propose an algorithm and a software tool that can be used as a decision support system for assigning the victims of a disaster to hospitals and for scheduling emergency vehicles for their transportation. Even though our algorithm could be used to handle daily ambulance responses and routine emergency calls, we target specifically a disaster scenario where the number of victims and the scarcity of the means of transportation are usually overwhelming. Indeed, while for normal daily operations the ambulances can be sent following the order of the arrival of emergency calls, when a disaster happens this First In First Out policy is not more acceptable. In these cases, the number of victims involved and the quantity of damages require a plan and a schedule of rescue operations, where usually priority is given to more critical cases, trying in any case to maximize the number of saved persons. In this context there are clearly also essential ethical issues which we do not address in this paper (for example, is ethically acceptable not to save a person immediately if this behavior allow us to save more persons later on?).

Our tool then assumes a simplified scenario where the number, the position and the criticality of victims is known.

The tool computes solutions that try to maximize the global number of saved victims. In many practical cases finding the optimal solution in not computationally feasible, hence we use a relaxation of the pure optimization problem. Our approach uses a *divide-et-impera* technique that exploits both Mixed Integer Programming (MIP) and Constraint Programming in order to solve the underlining assignment and scheduling problems.

To evaluate the effectiveness of our approach we have compared it against two alternative approaches: on one hand, a greedy algorithm based on the heuristic that send the ambulances first to the most critical victims and later to the others and, on the other hand, a complete algorithm that tries to find the optimal solution in terms of number of rescued victims.

Empirical results based on random generated disaster scenarios show that our approach is promising: it is able to compute the schedule usually in less than half a minute and almost always save more victims than other approaches.

Paper Structure. In Section II we define the model we are considering while in Sections III and IV we present the algorithms and the tests we have conducted. In Section V we present some related work. We conclude giving some

DOI: 10.9781/ijimai.2013.248

directions for future work in Section VI.

II. Model

In the literature a lot of models have been proposed to abstract from a concrete disaster scenario. Some of them are extremely complex and involve a lot of variables or probability distributions [3], [4]. For the purposes of this paper we adapt one of the simplest models, following [5], which considers only three entities: victims, hospitals, and ambulances. However, note that the flexibility of the Constraint Programming paradigm would also allow to handle more sophisticated models. Formally, we consider three disjoint sets:

- the set of the ambulances $Amb := \{A_1, \ldots, A_m\}$;
- the set of the victims $Vict := \{V_1, \ldots, V_n\}$;
- the set of the hospitals $Hosp := \{H_1, \ldots, H_p\}$;

and we assume to know the following data:

- the spatial coordinates $sa_i \in \mathbb{R}^2$ of every ambulance A_i;
- the spatial coordinates $sv_j \in \mathbb{R}^2$ of every victim V_j;
- the spatial coordinates $sh_k \in \mathbb{R}^2$ of every hospital H_k;
- the capacity $ca_i \in \mathbb{N}$ of every ambulance A_i;
- the capacity $ch_k \in \mathbb{N}$ of every hospital H_k;
- the estimated time to death $ttd_j \in \mathbb{N}$ of every victim V_j;
- the estimated dig-up time $dig_j \in \mathbb{N}$ of every victim V_j, *i.e.* the time needed by the rescue team to be able to rescue the victim as soon as the ambulance arrives on the spot;
- the a function $T : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{N}$ that estimates the time needed by an ambulance to move between two given points;
- the initial time $start_i \in \mathbb{N}$ an ambulance become available (an ambulance may be dismissed or already busy when the disaster strikes).

We are well aware that, especially in a disaster scenario, these data may be difficult to retrieve, imprecise and unreliable. Nevertheless our model can exploit these data to compute a first solution and then later, when the information become more precise, it can be rerun to improve the computed solution. Moreover, in order to get these information one can use the results of such works like [6], [7] that allow to esteem the time to death of a civilian or to find the best routes to reach the victims.

Assuming that all the above information are known, our goal is then to find as quickly as possible an optimal scheduling of the ambulances in order to bring the maximal number of alive victims to the hospital. Of course, solving optimally such a scheduling may be computationally unfeasible, especially in the case of a large number of victims. Moreover, in our scenario, a fast response of the scheduling algorithm is important for different reasons. First of all, the quicker the response is, the faster we can move the ambulances and therefore more victims may be saved. In addition, waiting for a long time may be useless because usually information rapidly changes (*i.e.* more victims come, the criticality of the patients vary, the hospitals may have damages or emergencies, ambulances can be broken). Hence, spending a lot of time for computing an optimal solution that in few seconds could become non optimal may result in a waste of resources and then lead to the impossibility of saving some victims. On the other hand, a purely greedy approach that at each stage makes the locally optimal choice (according to heuristics such as the seriousness or the location of the victims) would be definitely faster, but could result in a smaller global number of victims saved.

III. Procedure

As previously mentioned, our aim is to find the best possible compromise between the optimality of the ambulances scheduling and the time it takes to find it.

For this reason, we propose an approach that at the same time allows to compute a solution within a reasonable time limit and still allows us to save more victims than greedy strategies. Motivated by the success of hybrid algorithms on problems of resource assignment and scheduling [8], we developed a mixed approach that basically lies in the interaction of two phases: the *allocation phase*, in which we try to allocate as many victims as possible to ambulances and hospitals, and the *scheduling phase*, in which we compute the path that each ambulance must follow in order to bring the victims to the hospitals. In this section we first detail these two phases and then we show the pseudo-code explaining the interplay needed between the allocation and scheduling phase to solve the problem.

A. Allocation

In the allocation phase, we relaxed some constraints of the problem assuming that every ambulance can save in parallel all the victims it contains (in other terms, each ambulance with capacity *c* can be seen as the union of *c* distinct ambulances with capacity 1). The allocation of every victim to an ambulance and a hospital is performed by solving a Mixed Integer Programming problem by using two kind of binary variables, denoted by $a_{i,j}$ and $h_{j,k}$. The variable $a_{i,j}$ is set to 1 if and only if the victim V_j is assigned to the ambulance A_i, while $h_{j,k}$ = 1 if and only if the victim V_j is assigned to the hospital H_k. The constraints that we enforced are the followings:

- $\sum_{i=1}^{m} a_{i,j} \leq 1$ (for each *j* = 1, …, *n* a victim V_j can not be assigned to more than one ambulance);
- $\sum_{k=1}^{p} h_{j,k} \leq 1$ (for each *j* = 1, …, *n* a victim V_j can not be assigned to more than one hospital);

- $\sum_{j=1}^{n} a_{i,j} \le ca_i$ (for each i = 1, ..., m the maximum number of patients on an ambulance A_i must not exceed its capacity);
- $\sum_{j=1}^{n} h_{j,k} \le ch_k$ (for each k = 1, ..., p the maximum number of victims in a hospital H_k must not exceed its capacity);
- $\sum_{i=1}^{m} a_{i,j} = \sum_{k=1}^{p} h_{j,k}$ (for each j = 1, ..., n a victim V_j is assigned to an ambulance A_i if and only if V_j is assigned to an hospital H_k: there must not be *'dangling'* victims);
- $\sum_{i=1}^{m} \sum_{k=1}^{p} (start_i + T(sa_i, sv_j)) \cdot a_{i,j} + dig_j + T(sv_j, sh_k) \cdot h_{j,k} < ttd_j$ (for each j = 1, ..., n the time an ambulance A_i needs to reach a victim V_j, dig up and bring her to an hospital H_k is enough to save her).

Since the objective of the MIP problem is to try to maximize the number of rescued victims, we defined an objective function which takes into account both the seriousness and the location of the victims. Specifically, we require the maximization of the following objective function:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \frac{1}{\alpha_{i,j}} \cdot a_{i,j} + \sum_{j=1}^{n} \sum_{k=1}^{p} \frac{1}{\eta_{j,k}} \cdot h_{j,k}$$

where:

$$\begin{aligned} \alpha_{i,j} &:= (ttd_j - dig_j) \cdot (start_i + T(sa_i, sv_j)) \\ \eta_{j,k} &:= (ttd_j - dig_j) \cdot T(sv_j, sh_k). \end{aligned}$$

Recall that solving this problem does not necessarily mean to solve the overall problem: the solution found gives an esteem of the victims that could be saved and a preliminary allocation of every victim to an ambulance and a hospital. Indeed, since this is a relaxation of the original problem, it may be possible that not all the victims allocated to an ambulance may be saved. Anyway, it is worth noticing that the allocation guarantees that at least one victim for ambulance can be rescued. Also, there are no restrictions on the number of hospitals that an ambulance can visit.

B. Scheduling

Once the victims have been allocated by the first phase, the scheduling phase allows to define the path that each ambulance must follow in order to maximize the number of victims saved. After solving the above MIP we can assume that the allocation phase identifies a partition $\Pi := \{A_1^*, \ldots, A_m^*\}$ where for each i = 1, ..., m we define:

$$A_i^* := \{(V_j, H_k) : V_j \text{ is transported to } H_k \text{ by } A_i\}.$$

The ambulance scheduling for each ambulance A_i is then obtained by computing a *minimal Hamiltonian path* in a weighted and direct graph derived from A_i^*. Given such an A_i^*, let us consider the graph $\mathscr{G}_i := (\mathscr{N}_i, \mathscr{A}_i, \omega_i)$ where:

- the set of nodes $\mathscr{N}_i$ corresponds to a set of spatial coordinates, in particular each node represents either:
  - the initial position sa_i of the ambulance A_i;
  - the position of the victims sv_1, ..., sv_{n_i} that A_i transports;
  - the position of the hospitals sh_1, ..., sh_{p_i} that A_i visits;
- the set of arcs $\mathscr{A}_i \subseteq \mathscr{N}_i \times \mathscr{N}_i$ corresponds to the movements that A_i can do from one node to another and it is defined as follows:
  - $(sa_i, sv_j) \in \mathscr{A}_i$ (A_i can go to any assigned victim V_j from its initial position);
  - if V_j is assigned to H_k, then $(sv_j, sh_k) \in \mathscr{A}_i$ (A_i can bring a victim to its assigned hospital);
  - if $V_j \neq V_{j'}$ are assigned to the same hospital, then $(sv_j, sv_{j'}) \in \mathscr{A}_i$ (A_i can move from an assigned victim to another one, but no victims assigned to different hospitals can be simultaneously on A_i);
  - if V_j is not assigned to H_k, then $(sh_k, sv_j) \in \mathscr{A}_i$ (A_i can move from an hospital to a victim only if she is not assigned to such hospital);
  - no other arcs belongs to $\mathscr{A}_i$ (no other move is allowed).
- the weight function $\omega_i : \mathscr{N}_i \times \mathscr{N}_i \to \mathbb{R}$ corresponds to the estimated time for moving from one point to another, including dig-up time:
  - $\omega_i(sa_i, sv_j) := start_i + T(sa_i, sv_j) + dig_j$
  - $\omega_i(sv_j, sh_k) := T(sv_j, sv_k)$
  - $\omega_i(sv_j, sv_{j'}) := T(sv_j, sv_{j'}) + dig_{j'}$
  - $\omega_i(sh_k, sv_j) := T(sh_k, sv_j) + dig_j$

Therefore, if A_i has assigned n_i victims and has to visit p_i hospitals, the number of nodes will be $|\mathscr{N}_i| = 1 + n_i + p_i$ while the number of arcs will be $|\mathscr{A}_i| = n_i^2 + n_i \cdot p_i$.

The scheduling of each ambulance A_i can be computed by finding the minimum cost Hamiltonian path $P_1 \to P_2 \to \cdots \to P_{n_i}$ in $\mathscr{G}_i$ where:

- P_1 corresponds to the initial location sa_i of A_i;
- P_j for each $1 < j < n_i$, corresponds either to the location of a victim or the location of an hospital;
- P_{n_i} is the location of an hospital of A_i^*;
- $\Omega < ttd_i$, where Ω is the total cost of the path and ttd_i the time-to-death of each victim of A_i^*.

The scheduling phase can therefore be mapped into a *Constraint Optimization Problem* (COP) with the goal of minimizing Ω and solved by using constraint programming techniques.

As already stated, it may be the case that not all the victims allocated to an ambulance may be saved, since differently to what happen in the relaxed problem now an ambulance has to save the victims sequentially. When this happens we have to compute a schedule that saves a maximal subset of such victims. However, instead of considering as maximal subset the one which contains the greater number of elements, we choose the one which has the maximum *priority value* that is calculated as follows. We first compute the remaining time $RT_i := ttd_i - dig_i$ of each victim by subtracting her dig-up time from the expected time to death. Then, given a subset of victims $W \subseteq Vict$, we set its priority to $w = \sum_{V_j \in W} \frac{1}{RT_j}$ (bigger values of *w* means higher priority). We decided to use this sum to evaluate the priority because, analogously to what happens for the harmonic average, the sum of the reciprocal gives priority to the victim having least remaining time and it mitigates at the same time the influence of large outliers (i.e. victims with big remaining time that can be easily saved later).

When an ambulance is scheduled, the model is updated accordingly and the allocation phase is possibly restarted in order to try to allocate the victims which have not yet been assigned. The procedure ends when no more victims can be saved.

C. A&S Algorithm

Listing 1: A&S Algorithm

```
1  ALLOCATE_AND_SCHEDULE($Amb$, $Vict$, $Hosp$):
2    while $Vict \neq \emptyset$:
3      REMOVE_NOT_RESCUABLE_VICTIMS($Vict$)
4      $\Pi$ = ALLOCATE($Amb$, $Vict$, $Hosp$)
5      sorted_ambs = SORT_AMBS_BY_PRIORITY($\Pi$)
6      foreach $A_i \in$ sorted_ambs:
7        $V$ = VICTIMS($A_i, \Pi$)
8        if $V \neq \emptyset$:
9          ($\Sigma$, $\Omega$) = SCHEDULE($A_i, V, \Pi$)
10         if $\Sigma$ is not feasible:
11           $w$ = 0
12           $\Omega$ = $+\infty$
13           foreach $k \in [|V|-1, \ldots, 1]$:
14             foreach $h \in [1, \ldots, \binom{|V|}{k}]$:
15               $V'$ = GET_SUBSET($h, k, V$)
16               $w'$ = $\sum_{v_l \in V'} \frac{1}{ttd_l}$
17               if $w' \geq w$:
18                 ($\Sigma'$, $\Omega'$) = SCHEDULE($A_i, V', \Pi$)
19                 if $\Sigma'$ is feasible $\wedge$
20                   (w' == w $\rightarrow$ $\Omega' < \Omega$):
21                   $w$ = $w'$
22                   $\Omega$ = $\Omega'$
23                   $\Sigma$ = $\Sigma'$
24         $start_i$ = $\Omega$
25         foreach $H_k \in Hosp$:
26           $ch_k$ = $ch_k$ − NO_OF_VICTIMS($\Sigma$, $H_k$)
27         $sa_i$ = LAST($\Sigma$)
28         $Vict$ = $Vict$ \ VICTIMS($\Sigma$)
29         schedule[$A_i$].append($\Sigma$)
30         if VICTIMS($\Sigma$) $\neq V$:
31           break
32   return schedule
```

The main procedure called Allocate & Schedule (A&S) and presented in Listing 1 takes as input the set of ambulances Amb, the set of victims $Vict$, and the set of hospitals $Hosp$. It consists of a cycle where, first of all, the victims that can not be saved (*i.e.* victims with a remaining time less than or equal to 0) are removed. This operation is performed by the external function REMOVE_NOT_RESCUABLE_VICTIMS at line 3.

The ALLOCATE function solves the MIP problem described in Section III-A and returns the allocation of every victim to one ambulance and one hospital. The ambulances are then sorted by the function SORT_AMBS_BY_PRIORITY according to the sum of the priority values of the victims assigned to each ambulance; in this way, the schedule of the ambulances which transport victims with higher priority is performed earlier.

The nested loop starting at line 6 is responsible to compute the schedule of all the ambulances. Considering the ambulance A_i, in line 7 the variable V is defined to be the set of the victims assigned to A_i. If V is not empty then SCHEDULE(A_i,V,Π) returns a possible schedule $\sum$ and its cost Ω for the ambulance A_i. This is done following the procedure described in Section III-B. If the schedule problem has no solution (line 10) then another solution that involves less victims is computed (lines 11-23). In particular, the priority of the set of the victims *w* is initialized to 0, while the cost of the solution Ω to $+\infty$ (lines 12-13).

The loops enclosed between lines 13 and 23 have the aim of calculating a maximum eligible subset of victims, *i.e.* a subset V' of V that both maximizes the value of $\sum_{v_l \in V'} \frac{1}{ttd_l}$ and admits a feasible schedule. In order to compute all the subsets of V we exploit the function GET_SUBSET(*i, k, A*) that returns the *i*-th subset (*w.r.t.* lexicographic order) among all the $\binom{|A|}{k}$ subsets of A with cardinality *k*. Note that computing all the subsets is in general exponential on the capacity of the ambulances. However, in real cases, this can be computationally feasible since the capacity of the ambulances is usually small. In line 15 a subset V' is retrieved and its weight is computed in line 16. In case the weight is greater than the weight of the current solution, a schedule of the ambulance A_i for victims in V' is computed (line 18). If the schedule is feasible and has a lower cost in case of equal weight then the current schedule $\sum$, the current weight *w* and the current cost Ω are updated.

By construction, once exiting from the above loops a schedule that involves at least a victim is always found. Then, we just need to update the model. First, we update the start time of the ambulance A_i with the total cost of the schedule $\sum$ (line 24). Then, for each hospital H_k we decrease the hospital capacity ch_k according to the number of victims that A_i brings to them (lines 25-26). The new spatial location of A_i is set to the value of the last hospital it visits (line 27) and we remove from $Vict$ all the victims that A_i has rescued. Finally, the schedule $\sum$ is added to the associative array *schedule* that is the output of the A&S procedure.

In case only a part of the victims allocated to the ambulance A_i were saved (line 30) the cycle starting at line 6 is interrupted in order to compute a new allocation that may allocate the remaining victims to other ambulances.

The algorithm terminates when no more victims can be saved. From the computational point of view this algorithm cyclically solves MIP and COP problems, which are well known NP-hard problems. However, by exploiting the relaxation of the MIP problem, on one hand, and the limited size of the COP problems, on the other, it is possible to get quickly optimal solutions by exploiting current MIP and COP solvers. An empirical proof of this is provided in the next section.

IV. Tests

We did not find in the literature suitable and extensive benchmarks of disaster scenarios that we could use to evaluate and compare the performances of our approach. For this reason, in order to evaluate our algorithm we extended the methodology used in [9]. In particular, we built random generated scenarios obtained by varying the number of hospitals in the set {1, 2, 4}, the number of ambulances in {4, 8, 16, 32, 64}, and the number of victims in {8, 16, 32, 64, 128, 256, 512}. The position of each entity was randomly chosen in a grid of 100×100 by using the Euclidean distance to estimate the time needed for moving from one point to another.

The capacities of the ambulances and the hospitals were selected randomly in the intervals [1..4] and [300..1000], respectively, while the dig-up time and the time to death of every victim were randomly chosen in [5..30] and [100..1000], respectively. For *i* = 1, ..., *m* we considered initially $start_i = 0$.

Listing 2: GREEDY Algorithm

```
1  GREEDY($Amb$, $Vict$, $Hosp$):
2    REMOVE_NOT_RESCUABLE_VICTIMS($Vict$)
3    sorted_victs = SORT_VICTS_BY_PRIORITY($Vict$)
4    while sorted_victs ≠ []:
5      $V_j$ = sorted_victs.pop()
6      $A_i$ = CLOSEST_AMB($V_j$)
7      schedule[$A_i$].append(($start_i$, $sv_j$))
8      $H_k$ = CLOSEST_HOSP($V_j$)
9      deadline = $ttd_j$
10     position = $sv_j$
11     victims = 1
12     time = $start_i + T(sa_i, sv_j) + dig_j + T(sv_j, sh_k)$
13     hosp_time = $T(sv_j, sh_k)$
14     foreach $V_l \in$ sorted_victs
15       t = $T($position$, sv_l) + dig_l + T(sv_l, sh_k) -$ hosp_time
16       if victims < $ca_i$ ∧ victims < $ch_k$ ∧
17         time + t < deadline ∧ time + t < $ttd_l$:
18         hosp_time = $T(sv_l, sh_k)$
19         sorted_victs.remove($V_l$)
20         schedule[$A_i$].append((position, $sv_l$))
21         deadline = $min$(deadline, $ttd_l$)
22         victims = victims + 1
23         time = time + t
24         position = $sv_l$
25     schedule[$A_i$].append((position, $sh_k$))
26     $start_i$ = time
27     $sa_i = sh_k$
28     $ch_k = ch_k -$ victims
29     REMOVE_NOT_RESCUABLE_VICTIMS(sorted_victs)
30   return schedule
```

To increase the accuracy and the significance of the results we tested our approach by running the experiments 20 times for each different scenario and by measuring the average number of rescued victims as well as the time required to solve the problem. In total we tested then 105 different scenarios.

For every different scenario we compared the results of A&S *w.r.t.* a greedy approach GREEDY and a complete approach COMP that are used as baselines. In the following, before reporting the results, we briefly explain the algorithms we used as baselines.

A. GREEDY

GREEDY is a heuristic based algorithm that at each time tries to assign the most critical victims to the closest available ambulance and then such ambulance to the closest available hospital. Moreover, GREEDY also looks if in the path from the ambulance to the hospital it is possible to save other critical victims.

The pseudo-code of GREEDY is summarized in Listing 2. As in Listing 1, the main procedure takes as input the ambulances, the victims, and the hospitals. After removing all the non rescuable victims (line 2), it sorts all the victims by decreasing priority (line 3). At line 4 it starts the main loop, which is repeated until no more victims can be saved. First, the most critical victim is extracted into the variable V_j (line 5): this is the victim that the ambulance will save first. In line 6 the function CLOSEST_AMB is used to retrieve the closest free ambulance A_i to victim V_j while CLOSEST_HOSP is a function that returns the closest available hospital to V_j. In lines 9-13 we define some auxiliary variables for computing the schedule of the ambulance. In particular `deadline`, `position`, and `victims` are used to store respectively the minimum time to death of the transported victims, the position of the last victim of the schedule, and the total number of victims on the ambulance. The variable `hosp` keeps track of the total cost of the path from the ambulance to the hospital while `hosp_time` represents instead the cost of the last segment of the path, i.e. time needed for moving from the last victim to the hospital H_k.

The ambulance A_i is immediately sent to hospital H_k unless there are other victims that can be saved along the way. In order to look for such additional victims, we use a loop that scans each remaining victim V_l of `sorted_victs` (line 14). Within the cycle, in line 15 we evaluate the cost `t` needed for carrying the victim V_l to H_k. If such a transportation is possible, that is the capacity of A_i and H_k is not exceeded and there is enough time for saving all the victims of A_i (lines 16-17) then in lines 18-25 we update the ambulance schedule by updating the corresponding variables.

When the foreach cycle terminates, all the victims that could be saved by A_i (according to the heuristic) have been considered and therefore the ambulance is sent to the hospital. Therefore, in lines 26-28 we update the start-time of A_i, its location and the capacity of the hospital. Finally, in line 29 we remove all the not rescuable victims and the while cycle starts again until no more victims can be saved.

B. COMP algorithm

COMP is an algorithm that maps the rescue problem into a COP and computes the schedule of every ambulance without a pre-allocation phase. COMP is a complete algorithm: it tries to maximize the number of rescuable victims and when it terminates with success it always returns an optimal solution.

COMP assigns to all the ambulances, victims, and hospitals an unique identifier. In particular all the ambulances of Amb have an identifier $i \in D_a := [0..m]$, all the victims of $Vict$ have an identifier $j \in D_v := [m+1..m+n]$ and all the hospitals in $Hosp$ have an identifier $k \in D_h := [m+n+1..m+n+p]$.

The schedule of ambulance A_i at its j-th *round* (where by round we mean a path from its starting point to exactly one hospital) was encoded with an array $R_{i,j}$ of integer variables indexed from 0 to $ca_i + 2$. We then defined $m \cdot n$ arrays containing the identifiers of the victims and the hospital that each ambulance should visit in sequence.

$R_{i,j}[0] \in \{i\} \cup D_h$ is the index corresponding to the location of the ambulance A_i at the beginning of the j-th round. Since in the first round the starting point of A_i is always *start*$_i$ and in the following rounds the starting point is always the location of a hospital, we have $R_{i,1}[0] = i$ and $R_{i,j}[0] \in D_h$ for $i = 1,..., m$ and $j = 2,..., n$.

$R_{i,j}[1],\dots,R_{i,j}[ca_i] \in \{0\} \cup D_v$ are instead the indexes of the victims that A_i can rescue. In case the ambulance round was not filled completely one or more elements of $R_{i,j}$ are set to 0, signaling for every element set to 0 that a place was not used.

The last two elements of the arrays contain the index of the hospital where the ambulance ends its round and the total cost of the round. Note that each $R_{i,j}$ definition also entails that a victim can not be assigned to more than one hospital and that the maximum number of victims on A_i must not exceed its capacity ca_i. Additional constraints are needed in order to achieve the soundness of the solution and to reduce the search space. In particular, we added constraints enforcing that:

- the total cost of each round $R_{i,j}$ has to be lower than the time to death of each victim of $R_{i,j}$;
- if $R_{i,j}$ does not save any victim then all the subsequent rounds will not save any victim;
- the maximum number of victims in a hospital H_k must not exceed its capacity ch_k;
- a victim can occur in at most one round.

The objective of the COP is maximize the number of rescued victims, *i.e.* maximize the cardinality of the disjoint union of the victims assigned to each round. In order to encode the problem, we used basic constraints (such as <, +, ...) as well as *global constraints* [10] (namely, *element*, *alldifferent_except_0*, and *count*).

As can be imagined, solving such a problem may consume too many resources. In order to conduct the experiments using scenarios of nontrivial size we imposed some limitations that in some cases may result in a loss of completeness. We first limited the number of rounds for each ambulance to the ratio between victims and ambulances whenever the number of victims was greater than the number of ambulances. Then, during the computation of the solutions, we have limited the use of the virtual memory to 50% of the total available space and set a timeout of 300 seconds keeping the best solution founded up to that time if no solution was proven optimal.

C. Results

Fig. 1 shows the average percentage of rescued victims obtained by using A&S, GREEDY, and COMP approaches. The x-axis values represent scenarios sorted lexicographically by increasing number of victims, ambulances, and hospitals (labels are omitted for the sake of readability, since each x-value is actually a triple of values).

Our approach is in average able to rescue the 87.08% of the victims (from a minimum of 18.02% to a maximum of 100%). Considering the median value, in half of the scenarios we are able to rescue more than 99.38% of the victims.

In only 2 cases (0.02% of the scenarios) GREEDY is better than our approach, while in only one case COMP is better than A&S. However, in these few cases the difference of saved victims is minimal (between 0.31% and 1.48%) while the gap between A&S and GREEDY or COMP can reach peaks of about 78% and 100% respectively. In average, A&S is able to rescue about 31.03% of victims more than GREEDY and 59.38% more than COMP.

From the plot we can also see that our approach is

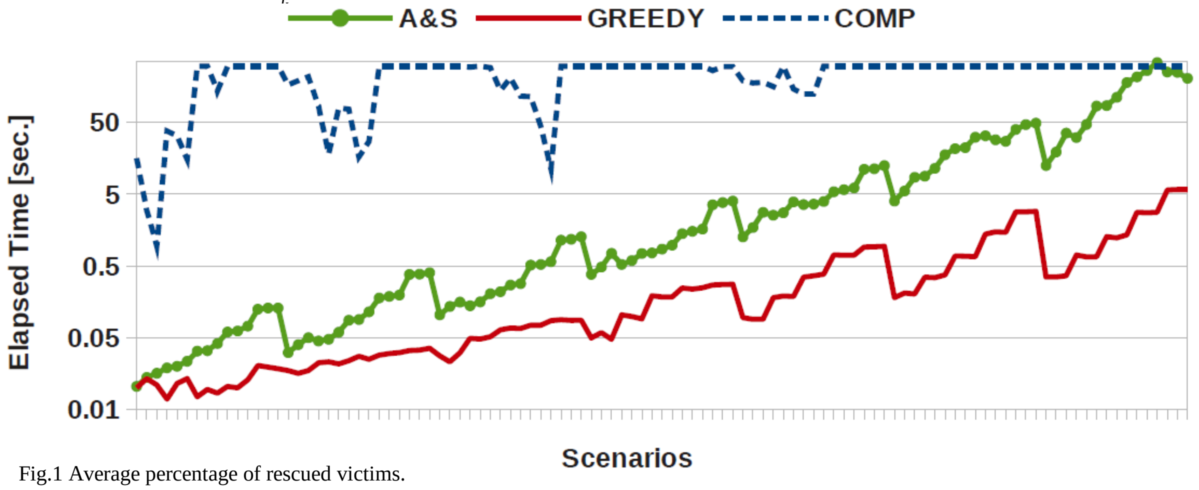

Fig.1 Average percentage of rescued victims.

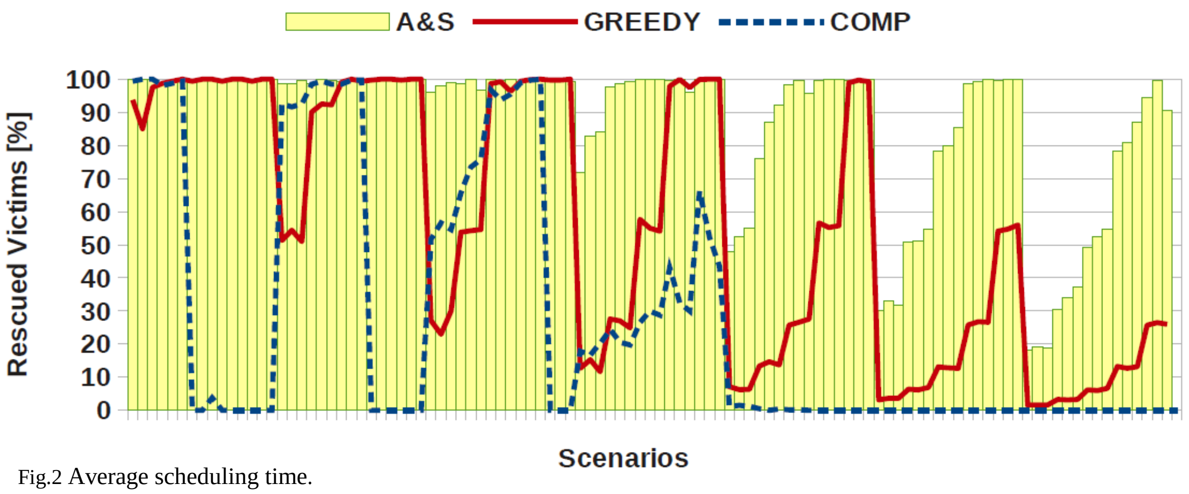

Fig.2 Average scheduling time.

especially better for scenarios involving a large number of victims. In particular, COMP algorithm can not find a solution within the timeout when the number of victims is greater than 128. It is therefore clear that this approach, although conceptually complete, is not scalable to large sizes. This can be a significant problem in scenarios like ours: it is not permissible to wait 5 minutes and having no solutions. GREEDY usually makes local choices that have a huge impact on the total number of the victims that could be saved. Although better than COMP for scenarios with many victims, the gap between GREEDY and A&S is significant. On the other hand, our approach in these cases tries to come up with a better global choice and therefore it can be far superior than a simple heuristic based approach.

In Fig. 2 we show the time needed to compute the entire schedule of the ambulances (please note the logarithmic scale). Although it is not surprising that GREEDY is faster while COMP is slower (the timeout expired often), it can however be observed that our approach takes reasonable times. In fact, in average the ambulances are allocated in 24.48 seconds, which means that on average in less than half a minute all the ambulances will be able to know the path that should be followed. Moreover, the median value indicates that for half of the scenarios the entire schedule is computed in less than 1.17 seconds.

Technical details. All the experiments were conducted by using an Intel® Core™ 2.93 GHz computer with 6 GB of RAM and Ubuntu operating system. The code for COMP and GREEDY algorithms was fully developed in Python. In particular, GREEDY algorithm exploits Gurobi [11] optimizer for solving the MIP problem and Gecode [12] solver for the COP problems. The code for the subsets generation is instead taken from [13]. Differently, for COMP algorithm we used Python for generating a MiniZinc [14] model solved by using the G12/FD solver of the MiniZinc suite. All the code developed to conduct the experiments is available at http://www.cs.unibo.it/~amadini/ijimai_2013.zip

V. Related Work

In the literature many techniques from operational research and artificial intelligence have been used to tackle different aspects of the disaster management problem. Most of the approaches are trying to develop and study pre-event solution to decrease the severity of the disaster outcome. As an example, in [15] the authors study the best allocation of deposits that allows to handle in the most efficient way the rescue operations in case natural disaster happens. In [4], the authors use MIP in order to schedule the operation rooms and the hospital facilities in case of a disaster. These paper however have a different goal from ours: we are not concerned with considering preventing measures that could allow to mitigate the consequences of future disasters. We are instead concerned with saving more victims, after the disaster happened.

There is also a large literature related to the problem of deciding the initial location of ambulances in order to decrease the average response time for ambulance calls. However, very few papers deal with the computation of the schedule for an ambulance. Some authors focus just on computing the best path for an ambulance toward the victim. For instance in [6] the authors use graph optimization algorithms in order to find a path for an ambulance. In our work we assume to have such a path and we are concerned with the problem of defining the order of the victims that an ambulance should pick up. In [16] the authors propose a routing algorithm for ambulances but, differently from our case, their model is probabilistic and it has been applied just for two small scenarios (*i.e.* scenarios containing few ambulances and victims).

In [5] the authors proposed the use of an interactive learning approach which allows rescue agents to adapt their preferences following strategies suggested by experts. The decision of the ambulance is based on a utility function incrementally improved through expert intervention.

Differently from our approach the authors here use an heuristic to dispatch the ambulances which rely on expert decision makers, while we rely only on optimization techniques.

In [17] is solved a task scheduling problem in which rescuing a civilian is considered as a task and the ambulances are considered as resources that should accomplish the task. The goal is to perform as many tasks as possible by using the Hogdson's scheduling algorithm to compute the solutions. Differently from our case, the authors considered here only the execution cost of the task and its deadline, ignoring important constraints such as the capacity of the hospitals and ambulances.

Combinatorial auctions are used in [18], [19], [20] to perform task allocation for ambulances, fire brigades and police forces. An ambulance management center is represented as the auctioneer while ambulances bid for civilians to save. Each free ambulance makes several bids and the auctioneer determines the winners using a Branch and Bound algorithm. A drawback of this approach is that it is difficult for bidders to estimate the cost for bids containing many tasks. Moreover, as pointed out in [21], if bidders bid on each and every possible combination of tasks the computation of satisfactory results is computationally expensive.

The authors in [22], [23] proposed a model based on a Multi-Objective Optimization Problem. They adjust controllable parameters in the interaction between different classes of agents (hospitals, persons, ambulances) and resources, in order to minimize the number of casualties, the number of fatalities, the average ill-health of the population, and the average waiting time at the hospitals. Then, they use Multi-Objective Evolutionary algorithms (MOES) for producing good emergency response plans. Their underline model is completely different from ours and we argue that is not very adaptable to deal with continuous changes and unexpected situations.

In [20], authors proposed partitioning the disaster environment in homogeneous sectors and assigning an agent to be responsible for each sector. Similarly, in [24], the city areas are partitioned and assigned to an ambulance. The number of clusters is determined by the size of the city. Such solutions could lead to unfair partitioning and inefficient assignment of agents to partitions. A more powerful partitioning strategy based on the density of blockades on the roads was used in [25]. This approach however requires a real-time information of the environment that is costly and sometimes difficult to retrieve.

Similarly to the GREEDY algorithm that we use as baseline, in [26] an heuristic is used to allocate the victims giving priority to the civilian with the highest probability of death. The shortcoming of this approach is that the cost of travel of the ambulance from one civilian to another could be very large; this could lead to a huge loss of lives if the size of the map is too large as in real-world situations.

In [27] Earliest Deadline First algorithm is also used to form coalition for rescuing victims. The victim with earliest deadline is selected and the number of ambulances needed to rescue this candidate in time is computed and called coalition. The use of coalition formation however works well when a task cannot be performed by a single agent, which is not the case for the task of saving a victim.

Finally we are aware of the existence of commercial applications for Emergency Dispatching (*e.g.* [28], [29]). The technical details explaining how these software are working unfortunately are always missing.

VI. Conclusions

In this work we have described a procedure that can be used as a decision support tool for a post-disaster event when a big number of victims need to be transported to the hospitals.

The proposed algorithm takes into account the position of the victims and their criticality, and schedules the ambulances in order to maximize the number of saved victims. Even though there is no guarantee that the solution obtained is the optimal one, experimental tests confirm that the number of saved victims is greater than the one that could be obtained by using, on one hand, a greedy priority-based heuristic and, on the other hand, a complete algorithm with a reasonable timeout of 5 minutes. Moreover, the proposed solution is usually fast enough to assign all the available ambulances in less than half a minute.

As a future work it would be interesting to evaluate our approach in a more dynamic and realistic scenario since assuming that the whole model is known a priori is not very realistic. A dynamic approach needs to adapt itself to context changes (*e.g.* new incoming victims, ambulances out of service, the critical state of victims, etc...).

In the event of significant changes the solution may be quickly updated exploiting the current allocation of ambulances without recomputing from scratch everything.

Moreover we would like to integrate the model with heuristics developed by domain experts.

Adopting these heuristics will allow the system to be able to react to a change very quickly, by using a default behavior that later can be changed if a better solution is found solving the optimiztion problem.

Another direction worth investigating is to study the performance and the scalability of the algorithm proposed taking into account also the robustness of the solutions (i.e. how the solutions vary depending on small changes of the initial model).

References

[1] N. Altay and W. G. Green, “OR/MS research in disaster operations management,” European Journal of Operational Research, vol. 175, no. 1, pp. 475–493, 2006.

[2] S. Tufekci and W. Wallace, “The Emerging Area Of Emergency Management And Engineering,” Engineering Management, IEEE Transactions on, vol. 45, no. 2, pp. 103–105, 1998.

[3] G. Erdogan, E. Erkut, A. Ingolfsson, and G. Laporte, “Scheduling ambulance crews for maximum coverage,” JORS, vol. 61, no. 4, pp. 543–550, 2010.

[4] I. Nouaouri, N. Jean-Christophe, and J. Daniel, “Reactive Operating Schedule in Case of a Disaster: Arrival of Unexpected Victims,” in WCE, ser. Lecture Notes in Engineering and Computer Science. International Association of Engineers, 2010, pp. 2123–2128.

[5] T.-Q. Chu, A. Drogoul, A. Boucher, and J.-D. Zucker, "Interactive Learning of Independent Experts' Criteria for Rescue Simulations," J. UCS, vol. 15, no. 13, pp. 2701–2725, 2006.

[6] N.A.M. Nordin, N. Kadir, Z.A. Zaharudin, and N.A. Nordin, "An application of the A* algorithm on the ambulance routing," IEEE Colloquium on Humanities, Science and Engineering (CHUSER), 2011, pp. 855–859.

[7] S. A. Suarez and C. G. Quintero, and J. L. de la Rosa, "A Real Time Approach for Task Allocation in a Disaster Scenario," PAAMS, 2010, pp. 157–162.

[8] M. Lombardi, and M. Milano, "Optimal methods for resource allocation and scheduling: a cross-disciplinary survey," Constraints, vol. 17, no. 1, pp. 51–85, 2012.

[9] R. Amadini

[10] , I. Sefrioui, J. Mauro, and M. Gabbrielli, "Fast Post-Disaster Emergency Vehicle Scheduling," Distributed Computing and Artificial Intelligence, ser. Advances in Intelligent Systems and Computing, Springer International Publishing, 2013, vol. 217, pp. 219–226.

[11] N. Beldiceanu, M. Carlsson, S. Demassey, and T. Petit, "Global Constraint Catalogue: Past, Present and Future," Constraints, vol. 12, no. 1, pp. 21–62, 2007.

[12] "Gurobi - The overall fastest and best supported solver available," http://www.gurobi.com

[13] "GECODE - An open, free, efficient constraint solving toolkit," http://www.gecode.org

[14] J. McCaffrey, "Improved Combinations with the BigInteger Data Type," http://visualstudiomagazine.com/Articles/2012/08/01/BigInteger-Data-Type.aspx?Page=3, 2013.

[15] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack, "MiniZinc: Towards a Standard CP Modelling Language," in CP, 2007.

[16] T. Andersson, S. Petersson, and P. Värbrand, "Decision Support for Efficient Ambulance Logistics," ser. ITN research report. Department of Science and Technology (ITN), Linköping University, 2005.

[17] K. Ufuk, T. Ozden, and T. Saniye, "Emergency Vehicle Routing in Disaster Response Operations," in Proceedings of the 23rd Annual Conference on Production and Operation Management Society, 2012.

[18] S. Paquet, N. Bernier, and B. Chaib-draa, "Multiagent Systems Viewed as Distributed Scheduling Systems: Methodology and Experiments," in Advances in Artificial Intelligence, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, vol. 3501, pp. 43–47.

[19] S. Suarez and B. Lopez, "Reverse Combinatorial Auctions for Resource Allocation in the Rescue Scenario," in ICAPS Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems, 2006.

[20] B. Lopez, S. Suarez, and J. D. L. Rosa, "Task allocation in rescue operations using combinatorial auctions," in Proceedings of the sixth Catalan Congress on Artificial Intelligence. IOS Press, 2003.

[21] M. Sedaghat, L. Nejad, S. Iravanian, and E. Rafiee, "Task Allocation for the Police Force Agents in RoboCupRescue Simulation," in RoboCup 2005, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4020, pp. 656–664.

[22] Nair, Ranjit and Ito, Takayuki and Tambe, Milind and Marsella, Stacy, "Task Allocation in the RoboCup Rescue Simulation Domain: A Short Note," in RoboCup 2001, Springer-Verlag, 2002, pp. 751–754.

[23] N. Giuseppe, M. Venkatesh, N. Lewis, R. Dianne, T. Marc, H. Liza, P. Ian, and M. Bud, "Complexities, Catastrophes and Cities: Unraveling Emergency Dynamics," in InterJournal of Complex Systems, vol. 4068, no. 1745, 2006.

[24] G. Narzisi, V. Mysore, and B. Mishra, "Multi-objective evolutionary optimization of agent-based models: An application to emergency response planning," in Computational Intelligence, 2006, pp. 228–232.

[25] M. Nanjanath, A. J. Erlandson, S. Andrist, A. Ragipindi, A.A. Mohammed, A. S. Sharma, and M. Gini, "Decision and coordination strategies for robocup rescue agents," in SIMPAR, Springer-Verlag, 2010, pp. 473–484.

[26] S. Paquet, N. Bernier, and B. Chaib-draa, "Comparison of Different Coordination Strategies for the RoboCupRescue Simulation," in Innovations in Applied Artificial Intelligence, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3029, pp. 987–996.

[27] J. Habibi, S. H. Yeganeh, M. Habibi, A. Malekzadeh, A. Malekzadeh, S. H. Mortazavi, H. Nikaein, M. Salehe, M. Vafadoost, and N. Zolghadr, "Impossibles08 Team Description RoboCup Rescue Agent Simulation," July 2008.

[28] O. A. Ghiasvand, and M. A. Sharbafi, "Using earliest deadline first algorithms for coalition formation in dynamic time-critical environment," Education and Information Tech, vol. 1, no. 2, pp. 120–125, 2011.

[29] "Odyssey website," http://www.plain.co.uk/index.php?option=com_content&task=view&id=67&Itemid=98

[30] "GeoFES,website," http://www.dhigroup.com/MIKECUSTOMISEDbyDHI/GeoFES.aspx

Roberto Amadini received a Bachelor (2007) and Master (2011) degree in Computer Science from the University of Parma. He has worked as a Laboratory Assistant at I.I.S. "G. Romani" of Casalmaggiore (CR) from 2007 to 2009 while in January 2012 he started his Ph.D. program in Computer Science at the University of Bologna. Currently is a Ph.D. student working on theory and application of *Constraint (Logic) Programming* for modeling and solving combinatorial problems. He is also member of of the Focus Research Group at INRIA (France).

Imane Sefrioui received a graduate degree in Computer Science "Diplôme d'Ingénieur d'Etat" in 2011 from "Ecole Nationale des Sciences Appliquées" in Tangier. In January 2012, she started her Ph.D. program in Computer Science at the Faculty of Sciences of Tetuan in Abdelmalek Essaadi University. She is currently a Ph.D. student working on the techniques and algorithms for modeling and solving combinatorial optimization problems.

Maurizio Gabbrielli Maurizio Gabbrielli is professor of Computer Science at the University of Bologna and is director of the Ph.D. program in Computer Science. He received his Phd. in Computer Science in 1992 from the University of Pisa and worked at CWI (Amsterdam) and at the University of Pisa and of Udine. His research interests include constraint programming, formal methods for program verification and analysis, service oriented programming.

Jacopo Mauro received a bachelor and master degree in computer science from Udine University. In 2012 he receive a PhD in computer science from the University of Bologna. Since 2010 he is member of the Focus Research Group at INRIA (France). He has been involved in numerous Italian, French, and European research projects and a visiting student at CWI (Netherlands). He is currently working at INRIA and interested in Concurrent Languages, Service Oriented Computing, Constraint Programming, Constraint Handling Rules and AI Planning.