Capacitated Vehicle Routing Problem under Deadlines

Florent Dubois, Paul Renaud-Goud, Patricia Stolf

Capacitated Vehicle Routing Problem under Deadlines

Florent Dubois *University of Toulouse*
IRIT
Toulouse, France
florent.dubois@irit.fr
Paul Renaud-Goud *University of Toulouse*
IRIT
Toulouse, France
paul.renaud.goud@irit.fr
Patricia Stolf *University of Toulouse*
IRIT
Toulouse, France
patricia.stolf@irit.fr

***Abstract*—Fast floods are usually not predictable and lead to lot of damages. In the context of a fast flood, the rescue teams need to elaborate the most efficient plan to save people in the impacted area, "as fast as possible". Based on discussions with firefighters, the problem is formalized as a Capacitated Vehicle Routing Problem under Deadlines. We introduce tour planning which allows to plan vehicle trajectory for several travels through the rescue center to put casualties into safety. We model the "as fast as possible" requirement through two elements: (i) a deadline, *i.e.* the time before which someone has to get rescued, is associated with every demand, (ii) the objective function to minimize is the Flow-time, *i.e.* the sum over each victim of the period during which it was not into safety. We created a set of various graphs in order to evaluate our results varying size of the problem, temporal constraint parameter and capacity constraint parameter in order to observe the evolution of the model towards different kinds of problems. We express the problem as a MILP, which provides the optimal solution on reasonable instance size problems thanks to MILP solvers. Since finding the optimal solution in real-time will not be possible with MILP solver, we also propose and compare different heuristics algorithms. The results show that the heuristics results are close to the optimal solution given by the resolution using Linear programming formulation on small instances. The Best Flow-time Insertion algorithm shows better results than the other heuristics developed in this article for every problem size and it is the closest from optimal results for small size problems.**

***Index Terms*—Routing, Flooding, Capacitated, Deadlines**

I. Introduction

This article takes place in the context of the E-flooding ANR project which gathers multiple expertise in different domains around fast flooding problems. The goal of this project is to a give both a short and long term response to the flood. This article is a presentation of the model for the short term problem and its resolution. Flooding is a vast area of study in the Crisis Management domain because it results in important damages to infrastructures and danger for people. When some of the flooding can be predicted using meteorology's predictions, a certain category of floods called fast floods are not predictable and are characterized by a very quick increase of the water level in the affected area. In order to give the best response to that specific kind of crisis we need to optimize the rescue vehicles that will intervene at every point of the affected region where people need to be saved. To do so we need to solve an optimization problem known in the literature as Vehicle Routing Problem. The main issue of this problem is the impossibility to make plans or anticipate any actions so we need a real-time support decision tool for the rescue teams. Such problems are known as Dynamic VRP. The dynamism in this article is treated as a multi-period static problem which means we execute our model offline knowing the demands for the time-window and then repeat the process for next windows with new demands.

One of the main difference between our problem and widespread VRP is that unlike most of them, we are dealing with human lives and not merchandises. So where most of the existing VRP consider Time Windows for action to be made, we will consider **Deadlines** in the literal meaning which is the first contribution to the complexity of our model. Deadlines in this article are used as constraints where violation is not possible. In the opposite, in the literature the notion of time windows only implies a penalty on the objective function if not respected. In our problem, each demand has a release date which corresponds to the moment when a victim calls the rescue teams.

In addition to this hard constraint characterized by life or death stakes, we add soft constraints whom purpose is to improve rescue teams effectiveness. To do so we define the **Flow-time** as the time between the reception of a demand (made by the victims) and its treatment by the rescue teams. This leads to write an objective function to minimize the sum of Flow-times, weighted by their priority, for all demands. Furthermore we are

The work presented in this paper has been funded by the ANR in the context of the project i-Nondations (e-Flooding), ANR-17-CE39-0011

978-1-7281-4920-2/19/$31.00 ©2019 IEEE

looking for global optimal solution to our problem but in the knowledge of the experience's feedback from the firefighters we are associated with on E-Flooding project, we know that in real case situation, the crisis cannot be solved by a single tour of the rescue vehicles due to limited resources. That is why in our model we will consider several **Tours** for the rescue vehicles between the rescue center and the demand points.

We formalize the problem via linear programming: on the one hand, expressing this problem into constraints (in the form of inequalities) associated with an objective function makes the problem clearly defined. On the other hand, it facilitates the computation of an optimal solution on small-sized inputs thanks to linear programming solvers, such as Gurobi. We solve it on a wide set of graphs each simulating a crisis configuration based on SDIS 31 experience's feedback. The SDIS 31 is the firefighting organization of Toulouse's department. They are among others in charge of people relief in case of flooding and collaborate with us on the e-Flooding project. They gave us for instance logs from previous crisis for us to base our graph generation on. We were then able to build experiences based on real-life scaled demand size or travel times between nodes. Due to the complexity of this kind of problems, we cannot compute optimal solution in an affordable timing for big-sized inputs. As a consequence, we design heuristic algorithms to get a solution (though not optimal) in acceptable time. This double approach using MILP and heuristics also allows us to compare the results of our heuristics to optimal results on small-sized inputs.

The remainder of this paper is organized as follows. In Section II we present the literature around the problem studied. The Section III describes the problem and explains the mathematical model. The Heuristic algorithms are presented in Section IV and the computation results are detailed in Section V. We conclude in Section VI where we open the perspectives of our work.

II. State of the art

The Vehicle Routing Problem has been studied under different formulations in the literature. First it has been studied mainly in a static version in the 1980's and 1990's as reviewed in [1]. And thanks to the improve in technologies the subject has been revived under its dynamic version. Due to the nature of our problem, we need to consider a multi-period static problem which means we execute our model offline in knowledge of the demands for the current time-horizon and repeat this process with new demands on the next time-horizon making our model partially dynamic according to the definition of [2].

[3] presents a wide view of the applications that have been made in the area of Dynamic VRP especially in the area of emergency vehicles dynamic allocations. [4] solves VRP with uncertainty on demands by submitting Reliable, Robust and Fuzzy Selective VRP models extending [5]. [6] anticipates capacity limitation of vehicles and minimize the detour for restocking on an online model. We focus our studies on articles dealing with disaster relief situations such as [7] aiming at dispatching research and rescue teams and define dynamic problems as multi-periodic static problems which means that we could focus on a static model and expand it to a dynamic form later. Studying dispatching, [8] and [9] develop full dynamic models for ambulance relocation. It is focused on the anticipation to an event optimizing the response time of the rescue teams as well as [10] that uses a 3 layers approach to plan the rescue in disaster relief. [11] also presents a three-stage disaster relief plan considering uncertainty on infrastructure's state but the approach is focused on a delivery aspect and does not consider pickups. Furthermore in the case of flash flooding where by definition the anticipation phase is not applicable, the notion of time window presented in these articles could be adapted to fit the requisite. The work in [12] proposes a dynamic model adapted from the Pickup and Delivery Problem (PDP) with time window that could easily be adapted to our CVRPD. Furthermore this model considers uncertainty on both demands and travel times while trying to optimize equipment use. In the same category of problems, [13] tries to solve the problem using a Particle Swarm Algorithm and [14] showed that using heuristics is most of the time necessary in dynamic problem. With a different approach on exact methods [15] arrives to the same conclusion that we need to use heuristics if we want to solve a VRP in real-time.

The solution promoted by [16] keeps in the model all the information without simplifications with even uncertainty on demands. In order to be consistent with the real-time constraint it proposes to artificially reduce problem size using data fusion. The algorithms presented by [17] partially use spatial clustering as well as other neighborhoods local search in their VMND algorithm. For our purpose, this heuristic could be merged with [18] heuristic to handle dynamic problems re-launching the solver on appropriate chosen time intervals.[19] multi-scenario approach for dynamism is more suited to our requirements than [20] double-horizon approach for the degree of dynamism of our problem.

[21] uses parallel computing in order to speed up the resolution time in case of real-time models. [22] adapted the Pickup and Delivery Problem dynamically from a static version.

III. Problem description and mathematical model

In this section, we present the Capacited Vehicle Routing Problem under Deadlines (CVRPD).

We apply the CVRPD to people rescue in flash flooding's emergency phase, optimizing routes for rescue vehicles. We characterize our problem as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the vertex set $\mathcal{V} = \{0, \ldots, V\}$ of size $V+1, V \in \mathbb{N}$ where every vertex is a point of demand where people need to be rescued except the vertex 0 which is the rescue center. We use $\mathcal{V}^\star$ as the set of all demand points without the depot, and $\mathcal{E} = \{(i,j) : (i,j) \in \mathcal{V}^2, i \neq j\}$ the set of the direct edges representing existing roads that link nodes together. Each of these edges is associated with a cost reduced for our problem to a travel time tt_{ij}. Each demand i has a size d_i corresponding to the number of victims and has a time a_i for the action to be completed on the node (to rescue victims).

In the CVRPD we are looking for a global optimal solution of

a VRP problem for the current state of the graph. Due to the capacity limitations of the rescue vehicles and according to the SDIS 31 expertise, the rescue teams will not have resources to solve the problem with only one passage by the depot. We introduce the tours, indexed by $z \in \mathcal{Z} = \{1, \ldots, Z\}$ with Z the maximum number of times any vehicle has to go through the depot, we will then solve the problem on several tours. To deal with the capacity limit of the vehicles, we also need to consider that a vehicle $k \in \mathcal{M}$ has a maximum capacity Q_k, where $\mathcal{M}$ is the set of available vehicles. For the purpose of the model we also use x^z_{ijk}, a binary variable equals to 1 if and only if vehicle k visits vertex j using edge (i, j) in tour $z \in \mathcal{Z}$.

We study a Crisis Management case with people lives at stake so we need to determine for the model a way to differ the urgent nodes to be treated in priority from demands that do not need to be treated urgently. To do so, we based our categories of priorities on the ones of the fireman's department who use the following scale: (1) Can remain on the spot, (2) Have to be rescued within 12 hours, (3) Have to be rescued within 6 hours, (4) Need to be rescued in emergency. These 4 priority categories are used to characterize the problem with both priority factors and deadlines. For each node of $i \in \mathcal{V}^\star$ we will associate a deadline $f_i \in \mathbb{N}$ and a priority factor $p_i \in \mathbb{N}$.

While the hard deadlines cover the emergency aspect of the problem through the f_i's, the objective function minimizes the cumulative weighted time for the demands to be treated. We introduce the Flow-time which is the time between reception and treatment of a demand at node i: $h^z_{ik} - r_i$ for vehicle k on tour z with h^z_{ik} the absolute date of arrival of vehicle k to node i on turn z and r_i the release date of demand on node i which is the date we received the information of the demand. The objective of our optimization is to minimize the total Flow-time weighted by the priority for every demand. The aim of the optimization problem is to assign the demands to the vehicles and to the tours. For each vehicle on each tour we have to decide for a circuit of the graph going through the depot. For each node of each circuit we assign a part of the demands. We consider that the action time a_i is constant on a node even if only a part of the demands is assigned. We will consider that the first turn for the solution is for $z = 1$ and we will set all the variables for $z = 0$ to 0.

Using these variables and parameters we establish the following objective function:

$$\min \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} \sum_{k\in\mathcal{M}} \sum_{z\in\mathcal{Z}} (h^z_{ik} - r_i) \cdot p_i \tag{1}$$

subject to:

$$\sum_{k\in\mathcal{M}} \sum_{z\in\mathcal{Z}} q^z_{ik} = d_i, \quad \forall i \in \mathcal{V} \tag{2}$$

$$h^z_{ik} - r_i - R \cdot (1 - \sum_{j\in\mathcal{V}} (x^z_{jik})) \leq f_i, \quad \forall i \in \mathcal{V}^\star; k \in \mathcal{M}; z \in \mathcal{Z} \tag{3}$$

$$\sum_{i\in\mathcal{V}} x^z_{ijk} - \sum_{i\in\mathcal{V}} x^z_{jik} = 0, \quad \forall j \in \mathcal{V}; k \in \mathcal{M}; z \in \mathcal{Z} \tag{4}$$

$$\sum_{i\in\mathcal{V}} q^z_{ik} \leq Q_k, \quad \forall k \in \mathcal{M}; z \in \mathcal{Z} \tag{5}$$

$$h^{\phi(i,z)}_{ik} + a_i + tt_{ij} - R \cdot (1 - x^z_{ijk}) \leq h^z_{jk} \quad \forall i \in \mathcal{V}; j \in \mathcal{V}; k \in \mathcal{M}; z \in \mathcal{Z} \tag{6}$$

$$q^z_{jk}/Q_k \leq \sum_{i\in\mathcal{V}} x^z_{ijk}, \quad \forall j \in \mathcal{V}^\star; k \in \mathcal{M}; z \in \mathcal{Z} \tag{7}$$

$$\sum_{j\in\mathcal{V}^\star} x^z_{0jk} \leq 1, \quad \forall k \in \mathcal{M}; z \in \mathcal{Z} \tag{8}$$

$$\sum_{i\in\mathcal{V}^\star} x^{z-1}_{0ik} \geq \sum_{i\in\mathcal{V}^\star} x^z_{0ik} \quad , \forall z \in \mathcal{Z}^\star; k \in \mathcal{M} \tag{9}$$

With

$$\phi(i, z) = \begin{cases} z - 1 & \text{if} \quad i = 0 \\ z & \text{otherwise} \end{cases} \tag{10}$$

R an integer of big size compared to all other variables. We will use $R = 1000$ for instance

p_i	Priority: A constant coefficient used in objective function for demand i
f_i	Deadline: latest time for any vehicle to pick the last demand at node i
r_i	Release time: time when the demand i appears
d_i	Demand: The number of victims to rescue at node i
Q_k	Maximum capacity of vehicle k
tt_{ij}	Travel time from node i to node j
a_i	Action time for a demand at node i
R	High size coefficient
$\mathcal{M}$	Set of available vehicles
$\mathcal{V}$	Set of vertices in the graph
$\mathcal{V}^\star$	Set of demand points in the graph (without depot)

TABLE I
INPUTS

x^z_{ijk}	Binary variable equal to 1 if vehicle k use the edge from i to j during tour z
h^z_{ik}	absolute arrival time of vehicle k at node i on tour z
q^z_{ik}	victims taken by vehicle k at node i on tour z

TABLE II
VARIABLES

The objective function (1) is the sum of all the Flow-time for every intervention of vehicles, weighted by the priority factor of the demand.

Constraint (2) makes sure that solutions do treat every demands fully.

The constraint (3) is the deadline constraint that states that a solution cannot contain any completion time over the deadline associated with the demand. The third term is used to ensure the constraints only apply when node i is visited by vehicle k on turn z, in other cases the big factor R makes the inequality true for all reachable values of the other terms.

The (6) inequalities are necessary in order to ensure that Flow-times respect the timing imposed by travel times, action times compared to the previous interventions of a vehicle. Thereby the time of arrival at a node is equal to the time

```
l ⟵ sortDemands(criteria, demands);
while l not empty do
    while ∑ q_ik^z < ∑ Q_k do
        vehicle ⟵ firstAvailableVehicle();
        if noDeadlineViolation(vehicle,l[0]) then
            assignDemand(vehicle, l[0]);
        else
            nextVehicle();
        end
    end
    z ⟵ z + 1;
    EmptyVehicles()
end
```

Algorithm 1: GreedyDist and GreedySize algorithms

of arrival to the previous node to which we add the action time on the previous node and the travel time between these two nodes. Constraint (4) ensures that a vehicle that arrives to a vertex also leaves it and (5) set the maximum capacity of vehicles.

The quantity and binary variables q_{ik}^z and x_{ijk}^z are linked thanks to (7). (8) defines the tours as the route between two transitions through the depot and (9) makes sure there are no empty tours in the planning because it ensures that a vehicle that leaves the depot at tour z is also leaving the depot at tour $z-1$.

IV. Heuristics

The linear program expressed in the model has been solved; results will be presented in next section. In the context of our problem, we need a solution within a reasonable time. We however have to consider that the VRP is NP-Complete.

To be able to respect the solving timing we need to develop heuristics with the purpose to find the best solution possible without guarantee of optimum. We will discern heuristics integrated to the Linear Program and pure heuristics.

A. Linear Program Based Heuristic

The main issue for computation time is the size of the problem. To reduce it, we develop a Heuristic that launches the Linear Program several times on restricted sets of nodes. We first create a subset of nodes only containing demands of the highest priority. Once the problem is solved on this subset, we fix the assignments of the demands to the vehicles and extend the subset to the next priority. We follow this process until the fourth category. The result of this method will not be a global optimum but will be composed of local optimums. We call this algorithm DecreasePrio.

B. Greedy Algorithms

We develop algorithms that assign demands to vehicles. The first type of algorithms is an adaptation of the First Fit algorithm which is a classical algorithm for the bin packing problem. It scans previous bins in order and places the new item in the first bin that fits and starts a new bin only if it does not fit in any of the bins. The bins being here our

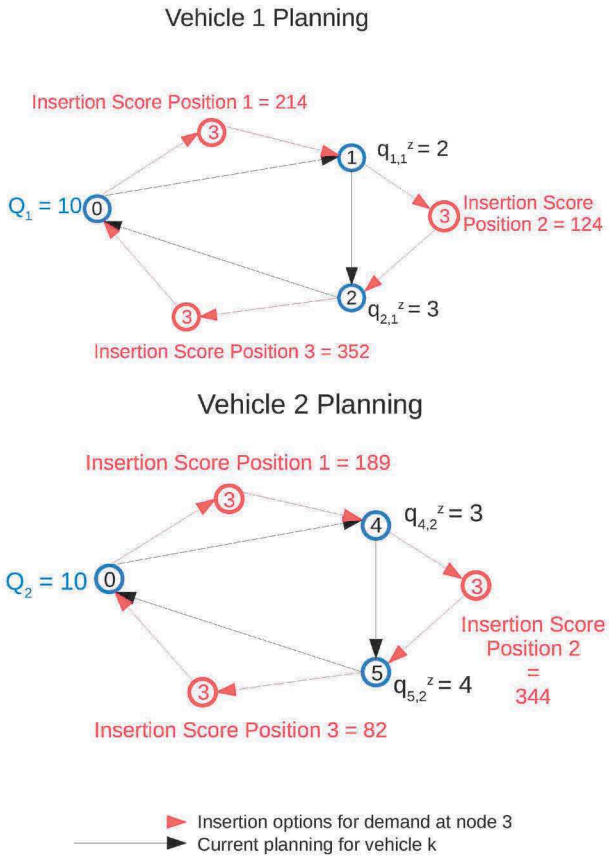

Fig. 1. bestInsertionScore example

vehicles and the items the victims to be rescued with the size of the demands or the shortest distance as a sorting criteria. We will use the function *firstAvailableVehicle* to get one random vehicle in the fleet that can handle the size of the demand. If none exists, the demand is split in several vehicles choosing the biggest available, and putting back the demand in the list with its size updated. For every assignment of the heuristics (expect for Greedy Alea) we check whether this assignment would violate a deadline. A failure of a heuristic then means that we have reached a situation in which we cannot insert the demand in any position of any available vehicle without violating a deadline. We then use *assignDemand* in order to update the current capacity of the vehicle. Finally, once a turn is completed we use *EmptyVehicles* in order to initialize the vehicles to their initial capacity translating the passage through the depot.

The Best Fit algorithm is another classical algorithm of the bin packing problem. Its principle is to place the new item in the spot such that the space left in the bin is minimal. The Best Flow-time Insertion (BFI) algorithm is an adaptation of the Best fit algorithm using the Flow-time used in the objective function as a decision criteria for the assignment. In the function *bestInsertionScore* we consider a vehicle and its current path (the nodes associated with the demand amounts that the vehicle carries in the current tour), and we do not modify the relative order of the nodes in the path. For each possible insertion of the current node/demand amount in this

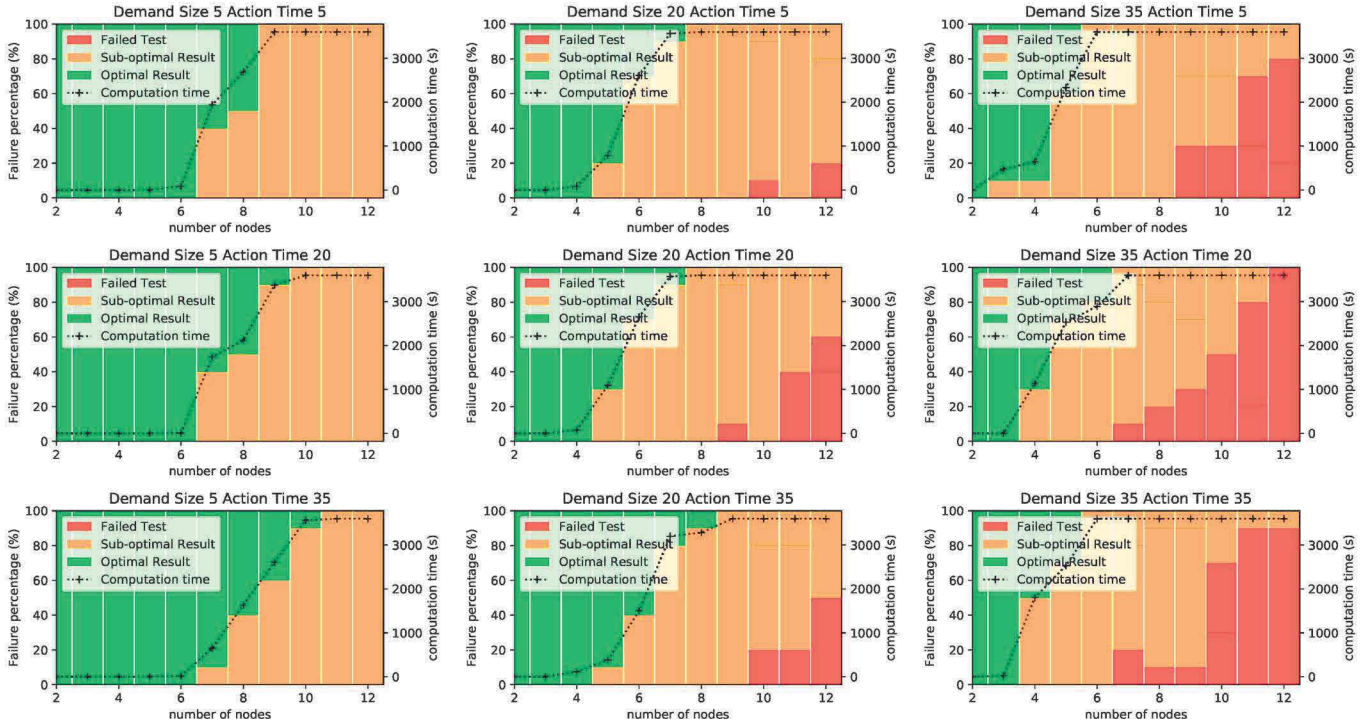

Fig. 2. Computation Time MILP

path, we compute the increase it would implicate on the objective function (the flow-time), We finally choose the position with the minimum contribution on the total weighted flow-time. If the action time is long the demand would probably be inserted at the end of the circuit; action time and travel time are taken into account to find the best trade-off. For example in fig. 1 we have two vehicles with the same initial capacity but different current planned routes. The *bestInsertionScore* function returns the pair with vehicle 1 and position 2 for this case. In fact the vehicle 1 has 5 places left and vehicle 2 only 3 so the best insertion score for vehicle 1 at position 2 is $124/5 < 82/3$. One can observe that a vehicle might not be able to deal with the whole demand; that is why we sort among vehicles by multiplying the flow-time increase by the demand amount that the vehicle can carry. The complexity of this algorithms is characterized by the complexity of the nested loop we can observe in the simplified algorithms. This complexity is $dmax * n$ in the worst case (if we rescue every person at every node one by one) with $dmax$ the size of the biggest demand point and n the number of nodes so the complexity of this algorithm is $\mathcal{O}(n^2)$.

$l \longleftarrow sortDemandsSize(demands)$;
while *l not empty* **do**
  while $\sum q_{ik}^z < \sum Q_k$ **do**
    $vehicle \longleftarrow firstAvailableVehicle()$;
    $(vehicle, position) \longleftarrow$
     $bestInsertionScore(l[0])$;
    if *noDeadlineViolation(vehicle,position,l[0])*
     then
      $assignDemand(vehicle, l[0])$;
    else
      $nextVehicle()$;
    end
  end
  $z \longleftarrow z + 1$;
  $EmptyVehicles()$
end

Algorithm 2: Best Flow-time Insertion Algorithm

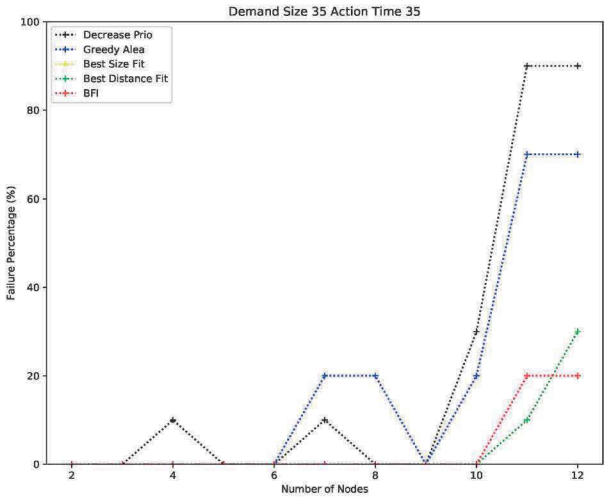

Fig. 3. Heuristics Failure

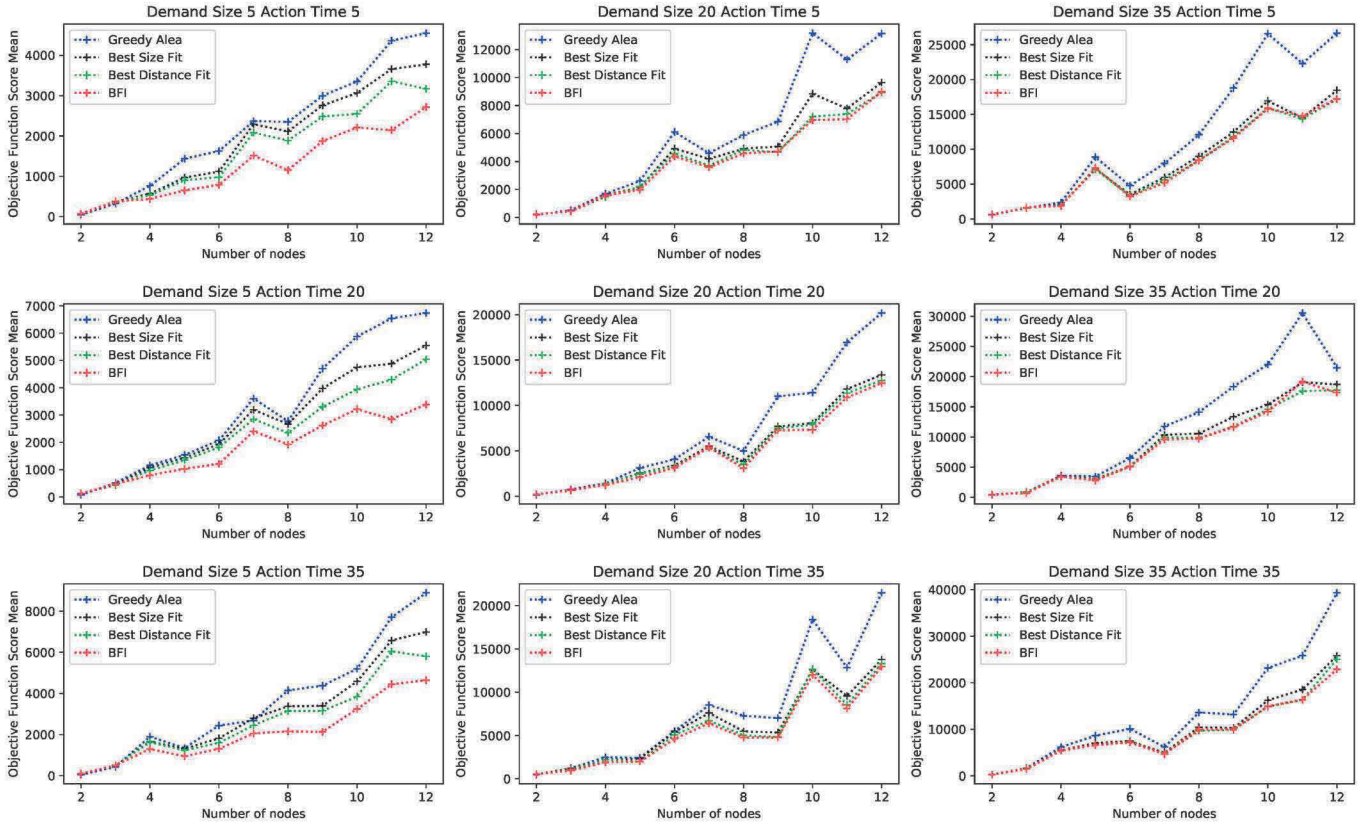

Fig. 4. Heuristics Objective Score

V. Experimental Results

In this section we describe the experimental process to evaluate our results. We want to evaluate the effect of different parameters on the model performances:

- Temporal constraint : action time evolution to test model reaction to closing deadline planing
- Complexity : demand size variation associated with capacity constraint evaluation
- Size of the problem : number of nodes of the graph

From these 3 parameters we created randomly generated graph with 10 graphs for every set of values. We fixed the number of vehicles to 10, their capacities to 10 and we assigned fixed priorities to demands. The number of nodes in the graphs will vary between 2 and 12 where we will have only three values for the action time mean and the demand size mean (5, 20, 35). The actual value for the action time and demand size will be a random distribution for every demand following a normal law for action time and an exponential law for demand size. This combination gives a total of 9 action time and demand size pairs, 11 graph size and for every of this triplets, 10 graphs for a total of 990 experiments. We used the GUROBI solver under version 8.0.1 on 8 parallel Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz Cores. Since the problem we want to solve needs to be solved as a real-time problem we consider that this is not useful to continue the computation after 1 hour for a given graph. So we stop the computation at 3600s even if the model did not find a solution or an optimum in this given time. This computation time limit is only applied to the MILP, the heuristics are all giving a result within the second. We first study the evolution of the computation time of the MILP according to the 3 parameters above. The graphics are displayed on a 3×3 grid where the Action Time is constant on a given row, while the Demand Size does not vary within a given column. Finally the Number of nodes is the abscissa parameter for every graphics. The computation time is displayed in fig. 2. For this experiment we considered 3 categories of results i) the MILP returns an optimal solution, ii) The MILP returns a solution but this solution was not proven optimal within one hour, we classify this as sub-optimal solution or iii) the MILP fails to return any solution within one hour, we consider this as a failure. On the y-axis we plot the solving time among instances on which the solver finds a solution within one hour. We could have considered one more category when the MILP detects an unfeasible problem but we never crossed this case in our experiment because we choose our parameters to avoid it.

We observe a clear breaking point in every graphics from which computation time starts increasing exponentially. This point appears at 7 nodes size problems for low constraints graphs down to 4 nodes size problems for the most constrained graphs. These results obviously show that the MILP reaches way to high computation time for real problems that SDIS 31 described. Furthermore these results also highlight the failure problem. In fact in the high constrained graphs the failure rate increase which translates the inability to get any

feasible solution through the MILP within one hour. All this considered, we use heuristics algorithms described above in order to find a feasible solution in short amount of time, the computation time is around the second for all heuristics. We then run the heuristics on the exact same set of graphs and get the results focusing at first on the failure aspect aiming at getting a solution whatever its quality. We plot the failure ratio of each greedy algorithms on a rather constrained problem (demand size: 35, action time: 35) in fig. 3.

We can see here in fact that two algorithms are better than the others, in fact Greedy Dist and BFI algorithms show lower failure rates than Greedy Size algorithm and the baseline algorithm Greedy Alea which assigns demands to vehicle randomly. On the Total of 990 graphs tested they only failed 6 times both. The results are satisfying in term of robustness of this algorithm, we observe a great benefit in term of failures compared to the MILP. We know we can get a solution in short time but the question left is the quality of these solutions. To evaluate them we keep the same algorithms and we display the mean of the objective function for the graphs where all 4 greedy algorithms found a solution in fig. 4.

Again in this graphic the curves show that in term of solution quality the same two algorithms (Greedy Dist and BFI) are better than the other with a slightly advantage for BFI for the low constrained graphs. We study the behavior of the to best heuristics on bigger graphs by increasing the number of nodes from 10 to 100. We choose the Demand Size and Action Time pair in order to avoid unfeasible problems.

Figure 5 confirms that BFI algorithm gives better quality solutions with a few failures on big graphs. Figures 6 and 7 show a comparison of BFI and MILP only when both get a solution.

As expected, the MILP is better than the BFI algorithm but the difference is reduced on highly constrained graphs. In fact on low constrained graphs the MILP reaches an optimal result more often than on highly constrained graphs which explains its efficiency compared to BFI, going up to 80% less efficient. But in highly constrained graphs where we can see the failures rate increases for the MILP, this difference is reduced to a maximum of 40%. These results confirm that the use a BFI as a heuristic for high size problem is a valid solution for our problem.

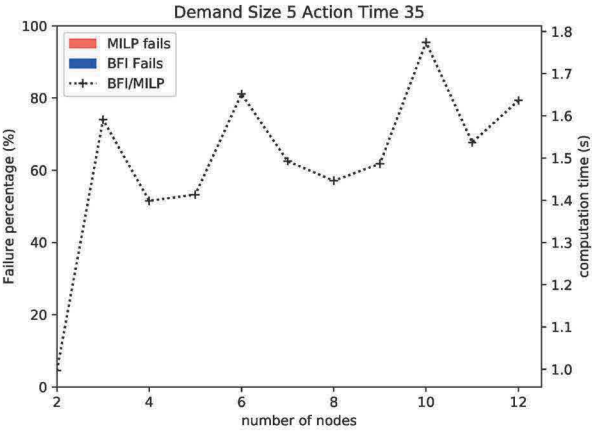

Fig. 6. Comparison MILP and BFI

VI. Conclusion

In this paper we studied the Capacitated Vehicle Routing Problem under Deadlines introducing the terminology of deadlines best suited for our problem of people relief in the Crisis Management context. We have developed a multi-tour MILP fitting our project requirement based on an objective function using the notion of Flow-time. We also proposed a set of heuristics including the Best Flow-time Insertion algorithm that we have compared to the MILP and validated thanks to experiments on a wide set of graph based on field experience of professionals of the field of people relief in case of flooding. We showed that the BFI was able to reduce the number of failure from 10,3% for the MILP to 0,6% keeping a maximum of 40% solution's quality lost for the Highly constrained graphs.

New research direction will be directed on using this heuristic mixed with the MILP in order to improve quality of the solution keeping the effectiveness in term of low failure rate of the BFI.

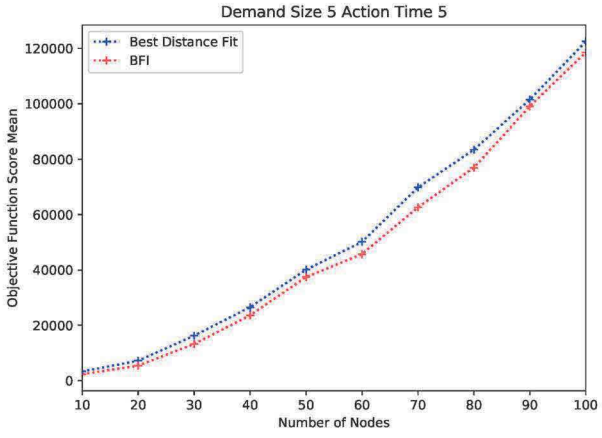

Fig. 5. Big Graphs BFI and Greedy Dist

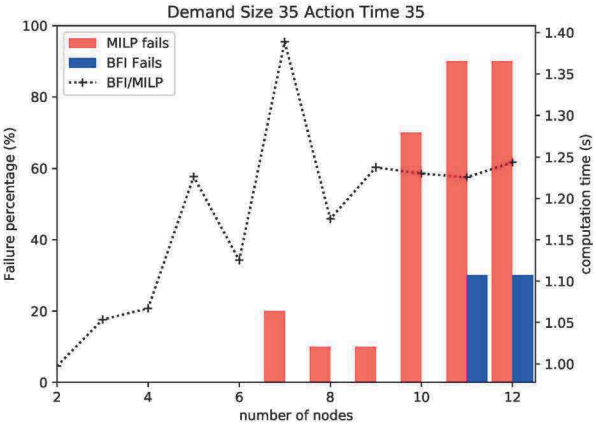

Fig. 7. Comparison MILP and BFI

References

[1] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European journal of operational research*, vol. 59, no. 3, pp. 345–358, 1992.

[2] A. Larsen, *The dynamic vehicle routing problem*. PhD thesis, Technical University of Denmark, 2000.

[3] V. Pillac, M. Gendreau, C. Guret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[4] M. Allahviranloo, J. Y. Chow, and W. W. Recker, "Selective vehicle routing problems under uncertainty without recourse," *Transportation Research Part E: Logistics and Transportation Review*, vol. 62, pp. 68–88, 2014.

[5] S. Ilgaz, O. Fernando, and MagedDessouky, "A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty," *IIE Transactions*, vol. 40, no. 5, pp. 509–523, 2008.

[6] W.-H. Yang, K. Mathur, and R. H. Ballou, "Stochastic vehicle routing problem with restocking," *Transportation Science*, vol. 34, no. 1, pp. 99–112, 2000.

[7] L. Chen and E. Miller-Hooks, "Optimal team deployment in urban search and rescue," *Transportation Research Part B: Methodological*, vol. 46, no. 8, pp. 984–999, 2012.

[8] M. Gendreau, G. Laporte, and F. Semet, "A dynamic model and parallel tabu search heuristic for real-time ambulance relocation," *Parallel Computing*, vol. 27, no. 12, pp. 1641–1653, 2001. Applications of parallel computing in transportation.

[9] L. Brotcorne, G. Laporte, and F. Semet, "Ambulance location and relocation models," *European Journal of Operational Research*, vol. 147, no. 3, pp. 451–463, 2003.

[10] J.-B. Sheu, "An emergency logistics distribution approach for quick response to urgent relief demand in disasters," *Transportation Research Part E: Logistics and Transportation Review*, vol. 43, no. 6, pp. 687–709, 2007. Challenges of Emergency Logistics Management.

[11] S. J. Rennemo, K. F. R, L. M. Hvattum, and G. Tirado, "A three-stage stochastic facility routing model for disaster response planning," *Transportation Research Part E:*

[14] M. Gendreau, G. Laporte, and R. Séguin, "An exact algorithm for the vehicle routing problem with stochastic demands and customers," *Transportation Science*, *Logistics and Transportation Review*, vol. 62, pp. 116–135, 2014.

[12] S. Wohlgemuth, R. Oloruntoba, and U. Clausen, "Dynamic vehicle routing with anticipation in disaster relief," *Socio-Economic Planning Sciences*, vol. 46, no. 4, pp. 261–271, 2012. Special Issue: Disaster Planning and Logistics: Part 2.

[13] B. F. Moghaddam, R. Ruiz, and S. J. Sadjadi, "Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm," *Computers & Industrial Engineering*, vol. 62, no. 1, pp. 306–317, 2012.
vol. 29, no. 2, pp. 143–155, 1995.

[15] D. Berkoune, J. Renaud, M. Rekik, and A. Ruiz, "Transportation in disaster response operations," *Socio-Economic Planning Sciences*, vol. 46, no. 1, pp. 23–32, 2012. Special Issue: Disaster Planning and Logistics: Part 1.

[16] A. Jotshi, Q. Gong, and R. Batta, "Dispatching and routing of emergency vehicles in disaster mitigation using data fusion," *Socio-Economic Planning Sciences*, vol. 43, no. 1, pp. 1–24, 2009.

[17] H. Larrain, L. C. Coelho, C. Archetti, and M. G. Speranza, "Exact solution methods for the multi-period vehicle routing problem with due dates," *Computers & Operations Research*, vol. 110, pp. 148–158, 2019.

[18] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Séguin, "Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries," *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 3, pp. 157–174, 2006.

[19] R. W. Bent and P. Van Hentenryck, "Scenario-based planning for partially dynamic vehicle routing with stochastic customers," *Operations Research*, vol. 52, no. 6, pp. 977–987, 2004.

[20] S. Mitrovi-Mini, R. Krishnamurti, and G. Laporte, "Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows," *Transportation Research Part B: Methodological*, vol. 38, no. 8, pp. 669–685, 2004.

[21] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno, "Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies," *European Journal of Operational Research*, vol. 151, no. 1, pp. 1–11, 2003.

[22] G. Berbeglia, J.-F. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, no. 1, pp. 8–15, 2010.