Received March 10, 2022, accepted April 14, 2022, date of publication April 25, 2022, date of current version May 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3170446

Capacitated Vehicle Routing Problem Under Deadlines: An Application to Flooding Crisis

FLORENT DUBOIS, PAUL RENAUD-GOUD, AND PATRICIA STOLF
IRIT Laboratory, University of Toulouse, 31330 Toulouse, France

Corresponding author: Florent Dubois (florent.dubois@irit.fr)

This work was supported by the French Research Agency under the Project e-Flooding (ANR-17-CE39-0011).

ABSTRACT Facing the issue of flash floods and their important damage both to victims and infrastructures, involved authorities are interested in the study of the ways to answer at best this kind of crisis both in the long-term and emergency phase. The victim relief operations can be optimized to help rescue teams improve their management of the crisis situation. It is translated into the field by the development of a decision support tool for victim relief operations. The problem addressed is a Vehicle Routing Problem (VRP) for the rescue vehicles. This article focuses on searching for efficient algorithms both in terms of execution time and intervention promptness to solve this VRPs. Data from a past crisis is used in this paper to evaluate the performances of the algorithms on problems as close to field experience as possible. Since rescue teams need to divide their forces into the different impacted sectors during flooding, algorithms to dispatch the resources (rescue team's vehicles) between areas of intervention are studied.

INDEX TERMS Floods, optimization, crisis management, vehicle routing.

I. INTRODUCTION

In the current context of climate change, disasters such as flooding are more likely to happen as stated in [12] and [34] for example. A response to these events is needed because it results in important damage to inhabitants and infrastructures. [35] offers a table of the impact of flooding in France from 1983 to 2010 in terms of deaths and estimated damages. Over this time period, a dozen of crises are referenced with at least 252 human casualties and more than 8,3 billion euros estimated damages each. As a part of the response, the crisis management field has been developed in the last years. This includes victim relief which is a life-saving challenge. In order to tackle this issue, we work directly with the SDIS 31 (Service Départemental d'Incendie et de Secours: French Firefighters that also intervene in case of floods) to help them fill the gap. In this article, we consider the short-term response's main problem: Victim relief. In order to give the best response, the routes for rescue vehicles that will intervene need to be optimized. This category of problem is called VRPs.

Our problem gravitates around VRP, and gathers elements from various flavors of VRPs

- Capacitated Vehicle Routing Problem (CVRP): Capacity limitations of the vehicles and quantities to be taken at demand points are considered.
- Split Delivery VRP: Sub-category of the CVRP where the demand points can be split between several vehicles as described by [17]. In our case, Split Delivery is mandatory because some demands cannot be served by a single-vehicle due to the number of victims.
- Heterogeneous VRP: This VRP allows to have vehicles of different capacities as presented in [22]. This is the case in our model where the rescue vehicles are not of the same capacities.
- Vehicle Routing Problem with Time Windows (VRPTW): In this category of VRP the demands need to be served after the beginning of the time window and before its end, as explained in [21]. The end of the time window is called the deadline.

Assembling all these types of VRPs and insist on the major aspects and difficulties of the problem, we call our problem Capacitated Vehicle Routing Problem under Deadlines (CVRPD). The decision was made to use the word Deadlines instead of the classical Time Windows terminology to distinguish this problem dealing with human life from commercial VRPs where Time Window violation does not have the same consequences. This term also places the problem more clearly in the crisis management context and more precisely in the victim relief area.

Flood crises impact the territory and victims at a different level often correlated with the water level. This means for the rescue teams different types of interventions that cannot be served using the same resources. These categories determine

The associate editor coordinating the review of this manuscript and approving it for publication was Alba Amato.

the types of demands that are in play but also the types of vehicles (and their associated average speed) that may be used for this kind of intervention.

VRP are known to be NP, it is to be determined whether this problem is NP-complete. In this case, the computation time of the solution would become a real challenge – at least for large instances. In fact, rescue teams work on short-term deadlines and cannot always be reached between two passages through the rescue center. Furthermore, we want to optimize vehicles routes over several interventions. That is why they try to anticipate future events using forecast services. Computation time must then be negligible relative to the time of completion of a tour in order to be consistent with a real-time decision support tool. However, NP-completeness in the strong sense problems are expected to become intractable when the problem size grows even for moderately small instances. Furthermore, we have observed in [18] that the Mixed Integer Program with Quadratic terms (MIQP) we developed indeed has exponential computation time, which explains the choice that was made to solve this problem through heuristic algorithms.

In this paper, we present the formulation of the model as a MIQP but as stated above, an exact method is not suited to answer the real-life problem in terms of computation time. However, this formulation allows characterizing a solution to the problem in order to confirm the results that heuristics algorithms produce and verify that the constraints of the model are verified. We present several algorithms in this article with the purpose of finding a solution in a short amount of time. They are based on an insertion strategy where the routes of vehicles are built gradually by inserting demands into it:

- Shortest Distance Insertion (SDI) algorithm: demands are inserted into the route of the current closest vehicle available;
- Best Flow-time Insertion (BFI) algorithm: demands are inserted to a vehicle, based on the lowest impact on objective;
- Best Flow-time Insertion with Order Questioning (BFIOQ) algorithm: same principle is applied as BFI adding a local search limited to the current route to improve the solution at every insertion.

In this paper, these algorithms are compared to a reference heuristic from the literature [33]. Hopefully, the solution developed in our work shows better performances in term of solution qualities and computation time than this baseline. This comparison also helps to highlight the advantages of our approach compared to other solutions from the literature to answer the characteristics of our problem.

Comparing with existing heuristics is not sufficient though. Data about victim relief operations is needed and its collection may be difficult in this context. In fact, for VRP in general the problem of study cases is a real challenge and a lot of studies choose to use instances from the literature or to generate random instances. In this paper, we generate instances based on data from past crises and use as much real data as possible knowing that some information will be missing if we want to replay a crisis identically since data logging is not a priority during the crisis. One of our study cases is the fast flood that occurred in Luchon's Valley in 2013. Thanks to the partnership with SDIS 31, which was in charge of the victim relief for this crisis, we were able to retrieve the Experience Feedback (EF) from this crisis that lists interventions of the rescue teams.

A flood can impact several areas at the same time forcing rescue teams to plan interventions simultaneously in these areas (we call them *sectors*). Using data extracted from EF, we have developed a process to generate the sectors. A flexible graph generator is important to randomly generate: (i) stakes on the impacted area, (ii) connectivity between stakes in the graph, and (iii) impacted stakes that turn into demands in a configurable way. Finally, we have addressed another of the problems encountered by rescue teams during a crisis through resources dispatch. In fact, rescue teams need to dispatch their resources between the impacted sectors. The geographic constraints impose to dispatch the resources of the rescue teams into several *sectors* in order to tackle the distance between demands and deal with several rescue centers. To help improve the response to the crisis, we have developed an algorithm in order to optimize resources dispatched through sectors with an application to a simulated crisis containing several sectors.

Several new contributions are presented in this article:

- Data extraction from Experience Feedback;
- A mathematical model of the problem to consider categories for the rescue vehicles;
- A heuristic Shortest Distance Insertion that mimics the behavior of the rescue teams;
- BFI and BFIOQ algorithms that are evaluated on field data;
- A configurable graph generator in order to evaluate algorithms on territory adapted from real conditions;
- An experiment based on the resources dispatch between sectors using the heuristics.

The remainder of this paper is organized as follows. In Section II we present the literature around the studied problem. The Section III describes the problem and explains the mathematical model. The heuristic algorithms are presented in Section IV and the Data Analysis that seeds problem instance generation is presented in Section V.

Experimental results are detailed in Section VI. We conclude in Section VII where we open the perspectives of our work.

II. STATE OF THE ART

VRP has been studied under various forms. First, it has been tackled as a static problem as reported in [25]. In 1959 already the subject was studied in [15] to optimally dispatch trucks of a fleet to serve customers. The VRP includes the Pick-up and Delivery Problem (PDP) where we might have to both pick up and deliver people as in [11] that uses heuristics and local search to optimize solutions of good distribution and waste collection. Reference [5] also works on carrier trucks minimizing route costs in a flexible delivery context. This article is an adaptation of the models from the literature to fit the multi-days delivery period of the study. Reference [13] focused on less-than-truckload delivery with the idea of sharing delivery vehicles between professionals that do not fully use vehicle capacity. This paper uses real data to validate the optimization of truckload and shipping synchronization. Recently [30] studied the VRP over a period of several days with the bi-objective of optimizing both route costs and driver consistency. A similar bi-objective can be found in the thesis of [7] where the objective is to minimize route cost and maximize planning stability for a garbage collection problem. Most of the VRPs study commercial problems and some of them are about same-day delivery where we search to optimize the routing of vehicles to deliver the order the same day of the command as in [29].

During a crisis, rescue teams need to deal with several categories of demands. A difference has to be made between interventions that need specialized vehicles, boats, or even helicopters. Furthermore for every and each of these resources of diverse categories, its capacity is a main constraint of the problem. In the work of [36], capacity limitations of vehicles are anticipated to minimize the detour for restocking (action of a vehicle when going back to the depot to refill or empty its load) on an online model. On the contrary, [10] studied a model for Pick-up and Delivery considering various categories of goods but did not consider capacity as a constraint. However, this article concludes that exact methods are not suited for problems of such complexity.

The studied problem is related to the split-delivery VRP. Reference [6] offers a review of the different problems of this VRP variation in the literature. For instance, [16] offers a heuristic for this problem but it is only fitted to problems where each demand is lower than the capacity of vehicles. Reference [3] offers a Tabu search to solve the problem but the computation time performances of such solutions are not appropriate to the Crisis Management Context.

The use of heuristic algorithms is then the best choice to answer this problem because its dynamic asks for a computation time kept under the minute. Contrary to some other VRPs, especially commercial applications, we need to keep a focus on the computation time of our algorithms. Indeed the degree of dynamism defined in [27] applied to our problem demands a computation time negligible compared to the time scale of the completion of an operation. Even if, the early article [20] tackled CVRP, the computation time of the experiments it presents is not adapted to a dynamic context. Different articles have studied heuristic solutions to the VRPTW. In [28], a comparison is offered with different heuristics from the literature. The comparison presents different high-quality solutions. For instance, [9] develops a heuristic based on a genetic approach. In this approach, two populations of solutions evolve simultaneously in order to improve solutions. The comparison also displays two-stage heuristics such as [8] and [23]. This approach generates routes on two-time horizons. In the short-term phase of the problem, the first solution is used to compute the route in a short amount of time. Then a search heuristic tries to improve the solutions on the long-term horizon. The comparison shows computation time is too important to fit the requirements of the crisis management context (over five minutes for the same scale of problems we want to study). These approaches use local search to improve the initial solution. Hence, to fit the computation time requirements induced by the context of our problem, a heuristic with a short computation time that does not contain an improvement routine seems more likely to produce feasible solutions in a short time. Reference [33] presents an insertion heuristic often used as a baseline in the literature. It computes solutions in computation time with the same scale we want to obtain. It also offers graph instances used as benchmarks for many papers from the domain. However, since these instances do not contain priorities, we will not be able to use them for evaluation.

This insertion algorithm is a reference however it does not consider priorities, which has to be considered in our problem. The quantity is also not considered in this heuristic. Finally one of the major drawbacks might come from the insertion, vehicle by vehicle, not comparing one vehicle with another. It will then be interesting to compare the results of our heuristic to this one. Reference [2] also worked on heuristics but to deal with online requests. Its application manages no capacity and routes over a day period. It is considering a short-term period and works on a look-ahead period to avoid to create infeasibility situation in the future. Recently [4] offered leads for local search in order to improve the quality of a first solution with a destroy and repair heuristic which deletes random demands from the route and tries to insert them somewhere else to improve the solution. Reference [26] also uses a local search algorithm as well as local clustering in the Variable MIP Neighborhood Descent algorithm. Working with a computation time constraint as well, [31] decides to use heuristics to find a solution in a time consistency with a re-optimization approach but on a single-vehicle problem.

The validation process for VRPs is elaborated either by replaying the events using collected data or simulating it based on a configuration the closest possible to the original problem. Reference [37] studies Hazard material transportation with an application area in Greece. The model is trying to minimize the impact of randomly generated incidents. Other studies validate their model by replaying a crisis like [32] with Taiwan earthquake on which they solved logistic material distribution. This article does not consider though capacity constraints as well as [14] that studies resources management and rescue team deployment. Another approach is presented in [1] that generates randomly 200 nodes size network to validate a humanitarian relief model adapted to the size of the specific crisis. Reference [24] used a high scale model of the 1994 earthquake in Los Angeles but with empiric probabilities on casualty types.

In this article, data from EF are used and instead of replaying the crisis identically, we choose to play a large-scale set of similar crises using the configuration obtained by data extraction. This choice is motivated both by the lack of data to replay the crisis identically and the objective to test the studied solution over different situations.

In relation to works from the literature, this article offers an application of the CVRP to a crisis management problem with data from a flash flooding event. It is built on the requirements of emergency teams. The article gathers specific VRP versions into one problem dealing with the capacity, deadline, categories, and dynamic aspects all at once. This article exclusively regroups the beforehand cited constraints and brings them into the domain of crisis management. It intends to tackle the problem of the computation time limitation in the context of emergency relief in order to meet rescue teams' expectations.

III. PROBLEM DESCRIPTION AND MATHEMATICAL MODEL

A. PROBLEM DESCRIPTION

In this section, we present the CVRPD adapted to handle several categories with shared resources.

We apply the CVRPD to people rescue in the flash flooding 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 $\mathcal{V} = [\![0, V]\!]$ of size $V + 1$, $V \in \mathbb{N}$, where every vertex is a point of demand where people need to be rescued. A person is considered rescued when it has been carried out to vertex 0, which is the rescue center. We use $\mathcal{V}^\star$ as the set of all demand points without the rescue center, 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 edge is associated with a cost, reduced for our problem to a travel time $tt_{i,j,c}$ where c the category of vehicle since travel speed depends on the vehicle. The category c is an integer in the set $\mathcal{C} = [\![1, nbCat]\!]$. 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 perspective of dealing with several categories with this model, it is necessary to introduce category variables. There are $nbCat$ categories that depend on the water level or type of intervention. Depending on the category of the demand, different types of vehicles are used and so the travel time also variate.

First, a parameter is needed to identify the category of a vehicle. For that purpose we define $cat_k \in \mathcal{C}$, for all $k \in \mathcal{M}$, which is the category of vehicle k. We also define the category of every demand i with $c_i \in \mathcal{C}$ where $i \in \mathcal{V}^\star$. This category might be subject to evolve over time depending on the water level for example. Since resources are specific to the type of intervention, categories are considered independent for this article. The categories are an input of the model given by the rescue teams. In fact, they can depend on other factors than the water level like the medical emergency of the situation for example. A really emergency demand might need an intervention by chopper for instance (category 4). Furthermore, the fifth category is dedicated to special interventions such as cattle evacuations for instance.

In the CVRPD we are looking for a global optimal solution to 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 may not have the resources to solve the problem with only one passage by the rescue center.

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 $x^z_{i,j,k}$, a binary variable which equals 1 if and only if vehicle k visits vertex j using edge (i, j) in tour $z \in \mathcal{Z}$. We introduce the tours, indexed by $z \in \mathcal{Z} = [\![1, Z]\!]$ where Z is the maximum number of times any vehicle has to go through the rescue center, we will then solve the problem on several tours.

We study a Crisis Management case with people's 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 base our categories of priorities on the ones of the fireman's department which uses 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 four priority categories are used to characterize the problem with both priority factors and deadlines. To each node $i \in \mathcal{V}^\star$ we then 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. In fact, we want to obtain a feasible solution, which translates into rescuing victims on time and also reducing at the minimum the waiting time for the victims to be rescued. We introduce the Flow-time which is the time between reception and treatment of demand at node i: $h^z_{i,k} - r_i$ for vehicle k on tour z with $h^z_{i,k}$ the absolute date of arrival of vehicle k to node i on tour z and r_i the release date of demand on node i which is the date we received the information of the demand. With the release date and deadline introduced in the model, the comparison can be made between CVRPD we are presenting and the Capacitated Vehicle Routing Problem with Time Windows highly represented in the literature, however, the terminology difference seems important to insist on the emergency aspect of CVRPD and to differ it from commercial problems for example. 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 on a circuit in the graph going through the rescue center. 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 tour for the solution is for $z = 1$ and we will set all the variables for $z = 0$ to 0.

B. MATHEMATICAL MODEL

We have seen in the state of the art section(II) that none of the existing works covers entirely the problem we deal with in this paper. Hence we describe here fully, and without ambiguity, the mathematical model, which embeds both the objective function to minimize and the constraints to fulfill. It relies on the constants and the variables that are summed up in tables 1 and 2. Using these variables and parameters, we establish the following objective function:

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

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

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

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

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

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

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

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

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

$$\sum_{i\in\mathcal{V}}(x^z_{i,j,k})\times c_j = cat_k,$$
$$\forall j\in\mathcal{V}^\star; k\in\mathcal{M}; z\in\mathcal{Z} \tag{10}$$

where

- $\phi(i,z) = \begin{cases} z-1 & \text{if } i=0 \\ z & \text{otherwise} \end{cases}$
- R is an integer of big size compared to all other 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 and the number of persons taken at this node.

Constraint (2) makes sure that solutions do treat every demand 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 tour z, in other cases the big factor R makes the inequality true for all reachable values of the other terms.

The inequalities (6) are necessary in order to ensure that Flow-times respect the timing imposed by travel times and action times compared to the previous interventions of a vehicle. Thereby the time of arrival at a node is equal to the time 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 at a vertex also leaves it and (5) sets the maximum capacity of vehicles.

The quantity and binary variables $q^z_{i,k}$ and $x^z_{i,j,k}$ are linked thanks to (7). (8) defines the tours as the route between two transitions through the rescue center and (9) makes sure there are no empty tours in the planning because it ensures that a vehicle that leaves the rescue center at tour z is also leaving the rescue center at tour $z-1$.

The constraint (10) has been added to the model to order the dependencies of the categories. This ensures that a vehicle can only be affected to a demand of the same category. This constraint guarantees that the categories are treated independently and could be treated as parallel problems.

In practice considering the complexity of the problem, we will not be able to solve it using exact methods and heuristic algorithms are a better fit to solve the problem in real-time.

TABLE 1. Inputs.

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
cat_k	Category of vehicle k
c_i	Category for node i
$tt_{i,j,c}$	Travel time from node i to node j for category c
a_i	Action time for a demand at node i
R	High size constant
$\mathcal{M}$	Set of available vehicles
$\mathcal{V}$	Set of vertices in the graph
$\mathcal{C}$	Set of integers for the categories
$\mathcal{V}^\star$	Set of demand points in the graph (without rescue center)

TABLE 2. Variables.

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

C. COMPLEXITY

The CVRPD is NP-complete in the strong sense.

Proof: We prove CVRPD is NP-complete by using a reduction from 3-Partition problem, known to be NP-complete in the strong sense [19]. A 3-Partition problem consists in deciding whether a set $\Gamma = \{b_1, \ldots, b_N\}$ of $N = 3n$ positive integers can be partitioned into n triplets

$\Gamma_1, \ldots, \Gamma_n$ (*i.e.* such that for any $k \in \{1, \ldots, n\}$, $\Gamma_k = \{g_{k,1}, g_{k,2}, g_{k,3}\}$) where $\sum_{i=1}^{3} g_{k,i} = B$. We will denote $\sigma : \{1, \ldots, n\} \times \{1, \ldots, 3\} \to \{1, \ldots, N\}$ the permutation such that for all $(k, i) \in \{1, \ldots, n\} \times \{1, \ldots, 3\}$, $g_{k,i} = b_{\sigma(k,i)}$.

First, CVRPD is NP since one can check in a polynomial time whether a given route is feasible or not. From any 3-Partition problem instance we call I_1, we build up an instance of CVRPD called I_2 as follows. In I_2, we dispose of n vehicles, *i.e.* $|\mathcal{M}| = n$, and the maximum capacity of the vehicles is set to 3. We consider a single category for this instance. We also consider a set $\mathcal{V}^\star = \{1, \ldots, N\}$ of N demands, whose action time is set to b_i: for all $i \in \mathcal{V}^\star$, $a_i = b_i$. Each node carries a single victim: $\forall i \in \mathcal{V}^\star$, $d_i = 1$. All the demands treated are from the same category numbered 1. The travel time for every edge of the graph $tt_{i,j,1}$ for all $i, j \in \mathcal{V}^\star$ is set to the same value of $2 \cdot B$. Finally the deadlines for every node are defined as follows $\forall i \in \mathcal{V}^\star, f_i = 9 \cdot B$. All the release dates are null: $\forall i \in \mathcal{V}^\star$, $r_i = 0$.

($\Rightarrow$) First we show that if there exists a solution to I_1 then there exists a solution to I_2. We assume that I_1 has a solution, *i.e.* there exist n triplets $\Gamma_k = \{g_{k,1}, g_{k,2}, g_{k,3}\}$, such that for any $k \in \{1, \ldots, n\}$, $\sum_{i=1}^{3} g_{k,i} = B$, and we build a solution to I_2. For every vehicle $k \in \{1, \ldots, n\}$ we use the sets Γ_k to provide a plan of the demands to be served. We have that $g_{k,i} = b_{\sigma(k,i)}$ and $b_j = a_j$ (by construction) for all $j \in \{1, \ldots, N\}$, hence the node $\sigma(k, 1)$ (resp. $\sigma(k, 2), \sigma(k, 3)$) needs an action time of $g_{k,1}$ (resp. $g_{k,2}$, $g_{k,3}$). We decide that vehicle k goes through node $\sigma(k, 1)$ then $\sigma(k, 2)$ then $\sigma(k, 3)$, dealing with the full demands. We remark that the capacity is not exceeded. Since all travel times are equal to $2 \cdot B$ by construction, the arrival date back at the rescue center for vehicle k is therefore: $2 \cdot B + g_{k,1} + 2 \cdot B + g_{k,2} + 2 \cdot B + g_{k,3} + 2 \cdot B = 4 \cdot 2 \cdot B + \sum_{i=1}^{3} g_{k,i} = 9 \cdot B$. All demands are satisfied and the deadline for every demand is fulfilled: we have exhibited a solution to I_2. In other terms, a solution of CVRPD is given by taking $q^1_{i,k} = 1$ for all $i \in \mathcal{V}^\star$ and $k \in \{1, \ldots, n\}$. We also need $x^1_{0,\sigma(k,1),k} = 1$, $x^1_{\sigma(k,1),\sigma(k,2),k} = 1$, $x^1_{\sigma(k,2),\sigma(k,3),k} = 0$ and $x^1_{\sigma(k,3),0,k} = 1$. For all $k \in \{1, \ldots, n\}$, $x^z_{i,j,k} = 0$ otherwise. The variable $h^z_{i,k}$ as to be affected according to the order of the plan determined by the $x^z_{i,j,k}$.

($\Leftarrow$) Now we show that if there exists a solution to I_2 then there exists a solution to I_1. We assume that I_2 has a solution. A vehicle can plan intervention to at most 3 nodes due to deadlines set to $9 \cdot B$ and the sum of travel times for 3 interventions equals $4 \times (2 \cdot B) = 8 \cdot B$. For the same reasons, the problem needs to be treated in only one tour. Otherwise for 3 interventions in 2 tours, the sum of travel times would equal $10 \cdot B$ ($6 \cdot B$ for the first tour and $4 \cdot B$ for the second one) and it would imply deadline violation. Since every node has to be rescued, and the total number of nodes is equal to $3 \cdot n$, a vehicle routes exactly 3 interventions. We define a permutation σ such that for all $k \in \{1, \ldots, n\}$ as an intervention on nodes $\sigma(k, j)$ for $j \in \{1, \ldots, 3\}$. With the deadlines $\forall i \in \mathcal{V}^\star f_i = 9 \cdot B$, by removing the travel times we have $\forall k \in \{1, \ldots, n\}$, $\sum_{j=1}^{3} a_{\sigma(k,j)} \leq B$. We have, for all $i \in \mathcal{V}^\star b_i = a_i$ consequently $\sum_{j=1}^{3} b_{\sigma(k,j)} \leq B$. In addition, knowing that $\sum_{i=1}^{N} b_i = n \cdot B$ we have that $\forall k \in \{1, \ldots, n\}$, $\sum_{i=1}^{3} b_{\sigma(k,i)} = B$. Therefore, I_2 has a solution if and only if I_1 has a solution.

Altogether, CVRPD is NP-complete in the strong sense. ■

IV. HEURISTICS

In this section, we will present the heuristics that we developed. First, we introduce SDI heuristics programmed as a reproduction of current rescue team behavior on the field. Secondly, BFI heuristic and its improved version are explained.

A. SHORTEST DISTANCE INSERTION ALGORITHM

This algorithm is based on the first-fit algorithm. The first fit is a resources allocation scheme. It is used in the bin packing problem where one must pack different size's items in bins also of different sizes. The list of items is sorted (in size order) and then items are allocated, in order, to the first bin in which they fit without consideration of the optimal choice. In our case, items are the nodes whose size is the number of victims and bins are the vehicles with their capacity. The list of nodes (demands) with Shortest Distance Insertion (SDI) is sorted first in terms of priority and then for the nodes of the same priority according to the distance of the closest available vehicle. When a vehicle is full it returns to the depot and is available for the next turn.

Detailed description of the algorithm is given in Figure1.

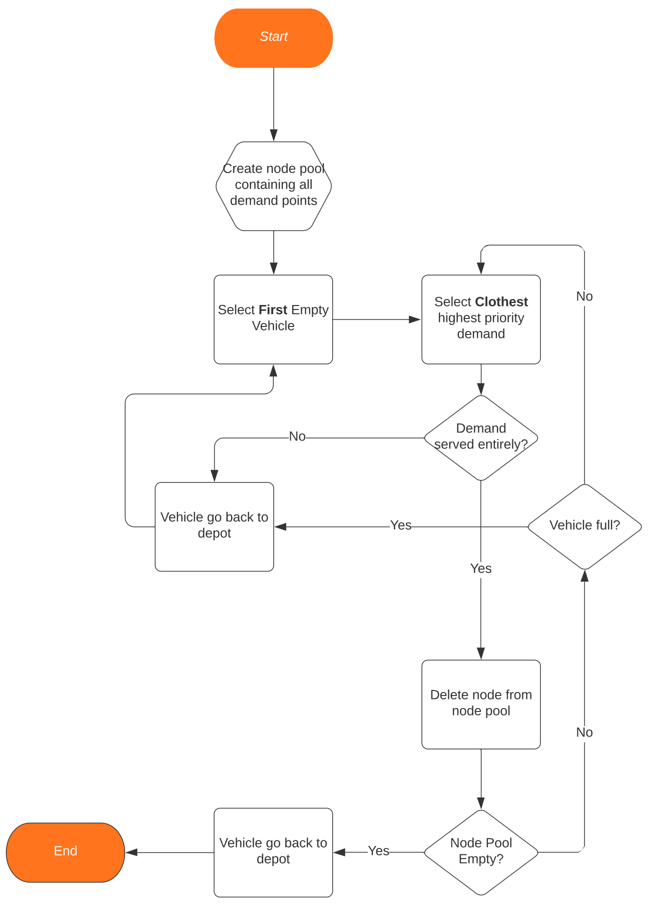
This heuristic represents the decision process of the rescue teams. However, we suspect it not to be the most efficient. In fact, the demand splits are not optimal since the quantity of rescued persons is not taken into account in the decision process in this algorithm. Furthermore, it is a greedy algorithm so the allocations are not re-assessed after a demand has been assigned to a vehicle. In fact, the process followed by rescue teams without decision support tools is to rescue the nodes by order of priority and then, using fast scooting techniques, assign rescue operation to the closest vehicle available. In this context, even if field experience can lead rescue teams operative to take an unpredictable decision, this algorithm is a good way to model rescue teams' relief decision process. From this observation, we will now be able to compare the rest of our heuristics to this one in order to evaluate the improvement rate it brings to the current situation.

B. BEST FLOW-TIME INSERTION ALGORITHMS

The allocation scheme of this heuristic is based on Best Fit. In opposition to First Fit, the purpose of this scheme is to allocate resources to the most appropriate task. The allocation process of this algorithm is described in algorithm 1.

With:

- *biggestDemand*: sorts the demands by priority and then by size in order to insert highest priority and biggest demand first



Start
Create node pool containing all demand points
Select First Empty Vehicle
Select Clothest highest priority demand
No
Demand served entirely?
No
Vehicle go back to depot
Yes
Vehicle full?
Yes
Delete node from node pool
No
End
Vehicle go back to depot
Yes
Node Pool Empty?

FIGURE 1. Shortest distance insertion algorithm flow chart.

Algorithm 1 Best Flow-time Insertion

```
$prio \leftarrow 1$
for $cat \in categories$ do
    while $prio < nbPrio$ do
        $dem \leftarrow biggestDemand(cat, prio)$
        for $vehicle \in \mathcal{M}$ do
            for $pos \in routeSize(vehicle)$ do
                $score \leftarrow FIS(pos, dem, vehicle)$
                if $score < bestScore$ then
                    $bestScore \leftarrow score$
                    $bestV \leftarrow vehicle$
                    $bestPos \leftarrow pos$
                end if
            end for
        end for
        $assignDemand(bestScore, bestV, bestPos)$
    end while
end for
```

- *routeSize*: looks for the different positions we might insert the new demand depending on the already built route
- *assignDemand*: assigns a demand to a vehicle. This operation modifies the solution's variables: $x^z_{i,j,k}$, $h^z_{i,k}$ and $q^z_{i,k}$ with k the vehicle of insertion, z its turn and for all the nodes i, j affected by the insertion.

The first step of this heuristic is to sort the demands by priority first and by decreasing size for demands of the same priority. The sorted list of the demands (updated each time a vehicle picks up partially demands) constitutes the queue line for the demands. Then for each demand, the Flow-time Insertion Score (FIS) is computed for every vehicle as follows for the vehicle k starting from node i for a rescue at node j: This score equals $+\infty$ if a deadline is violated by the insertion and otherwise:

$$FIS(i,j,k) = \frac{p_j \times (tt_{i,j,c} + a_i + h^z_{i,k})}{q^z_{j,k}}, \quad \forall z \in \mathcal{Z}, c \in \mathcal{C} \tag{11}$$

In this case the number of victims planned to be rescued at node j, $q^z_{j,k}$ is dependent on the current available capacity of the vehicle and it needs to be included in the score calculation in order to avoid the algorithm to allocate resources to treat demand points partially if it can be avoided. It does not forbid demand division but it reduces it compared to SDI. This FIS is compared between vehicles and the demand is allocated to the vehicle with the lowest score which means the lowest impact on the objective function. This process is then repeated as long as necessary to rescue all the victims. At the end of the loop on priority, we check if there is at least a vehicle left for the current tour, or else we change the tour: increment the current tour and empty the vehicles. Then we also check if there are demands left in the current priority, if not we switch to the next one. Finally, the same check is made on the categories at the end of the loop on categories. This algorithm terminates on 2 conditions:

- All demands are served. In this case, the algorithm terminates with a solution to the problem.
- For all left demands $FIS = +\infty$. The algorithm failed to find a feasible solution and generated a partial solution that violates constraints of deadline.

C. BEST FLOW-TIME INSERTION WITH ORDER QUESTIONING

This heuristic is based on BFI adding improvement. The principle of this algorithm is mainly the same as BFI but any time we add a demand in the route of the vehicle, the order of the demands that are already handled by this vehicle will be questioned, as shown in the next algorithm. A brute force algorithm is then launched. It builds all the $n!$ routes possible, with n the number of demands already in the route of the vehicle for this tour. Note that n is always lower than the capacity of the vehicle which makes the number of routes builds reasonable. An objective score for the tour is computed for each of these routes. The route with the lower score is selected and the assignment is made to the vehicle in the final order. During the computation of this score, capacity constraints do not enter in the calculation since they are not modified. The pseudo-code of this heuristic is presented in algorithm 2.

Algorithm 2 Best Flow-time Insertion with Order Questioning

```
prio ← 1
for cat ∈ categories do
    while prio < nbPrio do
        dem ← biggestDemand(cat, prio)
        for vehicle ∈ $\mathcal{M}$ do
            for pos ∈ routeSize(vehicle) do
                score ← FIS(pos, dem, vehicle)
                if score < bestScore then
                    bestScore ← score
                    bestV ← vehicle
                    bestPos ← pos
                end if
            end for
        end for
        assignDemand(bestScore, bestV, bestPos)
        for route ∈ routesSize(bestV)! do
            if OS(routes) < OS(bestRoute) then
                bestRoute ← routes
            end if
        end for
        assignNewRoute(bestRoute)
        checkTurn(z)
    end while
end for
```

With *OS* that computes the objective score of a given route. This function computes the value of eq 1 for a given vehicle and turn concerned by the local search limited to the current route.

D. SOLOMON HEURISTIC

In this section, a short presentation of the insertion heuristic presented in [33] is made. This heuristic is used as a baseline in the experimental part of our work. This heuristic starts by initializing every route according to a criterion: The farthest unrouted demand.

In this algorithm from literature, the vehicles are considered one after the other. Therefore, the routine initializes the route for a vehicle and then inserts demands in this route until the vehicle capacity is reached. Then it handles the next vehicle. When the route for the first tour is planned for all vehicles, the routine continues with the second tour for the first vehicle. Tours are incremented until all demands are served. The algorithm depends on parameters λ, μ, α_1 and α_2 that can be adjusted to adapt the performances of the algorithm by changing the weights of different factors. Further details about the values used for parameters λ, μ, α_1 and α_2 are given in sectionVI.

Once a route has been initialized for a vehicle, the heuristic tries to insert demands optimally in the route. At every insertion, 2 criteria are used:

- The first criterion c_1, is used to determine for each node, the best feasible insertion spot. It is based on a sum of two terms that represent the temporal deviation induced by the insertion of the node in the existing route and the delay in service for the next demands in the route. In fact, insertion is not necessarily at the end of the route but at different positions. It might lead to offsetting an already planned intervention to a node. Both terms are weighted by α_1 and α_2 respectively.
- The second criterion c_2, allows determining which demand is inserted knowing the results of best position selection with the first criterion. The demand selected is the one minimizing the difference between the travel time from the depot, and the first criterion is weighted by λ and μ respectively.

Solomon Insertion Heuristic pseudo code is presented in algorithm 3.

Algorithm 3 Solomon Heuristic

```
Input: d: List of demands
k ← 1
z ← 1
while d ≠ [ ] do
    initializeRoute(k, d)
    while $\sum_{i\in\mathcal{V}^\star} q^z_{i,k} < Q_k$ do
        positions ← computeBestPositions($c_1$, d)
        bestNode, pos ← bestNode($c_2$, d, positions)
        insertDemand(bestNode, pos, d)
    end while
    if k = length($\mathcal{M}$) then        ▷ Vehicles full in current tour
        z ← z + 1                            ▷ Switch to next tour
        k ← 1                                ▷ Select first vehicle
    else
        k ← k + 1                            ▷ Switch to next vehicle
    end if
    l ← sortDemands(criterion, d)
end while
```

E. RESOURCES DISPATCH

As stated earlier, a flood might impact several sectors. However, rescue teams might have to handle the sectors with their limited resources. We implement algorithms that dispatch the resources among sectors to optimize the response by minimizing the objective overall sectors. The first approach we follow in order to make the best resources dispatch is to try all the possible configurations and keep the best one. This is done by launching a heuristic with x vehicles, for $x \in [0, M_{cat}]$ with M_{cat} the maximum vehicle number of a category. Then we pick the vehicle dispatch solution for which the sum over all sectors of the Flow-time objective is the lowest. But this Brute Force Resources Dispatch (BFRD) method might be expensive in terms of computation time.

That is why we also develop Greedy Resources Dispatch algorithms (GRD). These algorithms give a resources (vehicles) dispatch configuration based on the ratio of a specific metric over the different sectors. The three versions of GRD we test are based on 3 different metrics:

- GRD1 dispatches the total number of vehicles multiplied by the ratio of nodes of the sector.
- GRD2 dispatches the total number of vehicles multiplied by the ratio of victims in the sector.
- GRD3 dispatches the total number of vehicles multiplied by the ratio of the cumulative distance from the rescue center to the nodes of the sector.

These algorithms are very fast compared to the brute-force approach but as they do not try different configurations we expect that these greedy algorithms might give infeasible vehicles dispatch, which means they might not allocate enough vehicles to one sector for the algorithm to be able to find a feasible solution (there might not even exist a solution at all).

We then try another approach in order to get a feasible solution and also reduce the number of computations tested by BFRD. After checking with the results from BFRD that variations of the objective are constant, a second algorithm can be developed on the principle of a brute force. But instead of testing all configurations Constant Objective Detection Resources Dispatch algorithm (CODRD) can stop the computation when we reach 2 identical objectives in a row. This exit condition can be stated because we confirmed that the variations of the objective with BFRD are constant and therefore if the objective does not vary with the addition of a vehicle then it will not vary with more. For the experiments, we will apply these algorithms to 2 sectors but it can be generalized to more. In fact, a study on 2 sectors is sufficient to compare the different Resource Dispatch algorithms and avoids using the algorithm on more sectors which would only increase the number of possibilities of dispatch and therefore the computation time.

V. DATA ANALYSIS

With the objective of reproducing a real-life experiment to validate the model and heuristics, data has been extracted from Experience Feedback from rescue teams of SDIS 31. SDIS 31 is responsible for the rescue operations for the Haute-Garonne Department in the South of France. We conceived the model to fit their logistics and therefore the priority categories that they usually use. The values of the priority coefficients are fixed to:

1) Can remain on the spot: 1
2) Have to be rescued within 12 hours: 2
3) Have to be rescued within 6 hours: 4
4) Need to be rescued in emergency: 10

These values have been arbitrarily picked to represent the relative importance of each priority category one from another. They can be adapted consequently to discussions with the rescue teams in order to fit the situation requirements. In 2013 a flash flood has occurred in the valley of Luchon in the South of Haute-Garonne. The information kept in the experience feedback documents is not sufficient to simulate an identical crisis. Nevertheless, we were able to extract useful data to base our experiment on. Using this data, the goal is to build several territories model as graphs that are similar to the Luchon crisis. These experimental graphs will be referred to as Luchon-like.

One of the aspects that was not retrievable is the connectivity between the nodes of the graph. Since we aim at reproducing similar graphs, there is then a need to build random graphs keeping the control on some known parameters of the territories we intend to study.

The extracted data characterization is described in the first part of this section. Then the graph generator that was developed for these experiments is detailed. Finally, experimental results are presented and interpreted in the last part of this section.

A. EXPERIENCE FEEDBACK

During a crisis such as flooding, interventions are very distinctive and need a response using appropriate resources. That is why rescue teams consider mainly categories of interventions. Following the same characterization system, we looked into the experiment feedback from Luchon for statistics for each of these categories during the crisis. The number of vehicles for each category is the same as the Experience Feedback description except for category 4 where we reduced the number of helicopters on purpose to make the problem harder to solve in this category.

Category 1

This category of intervention represents mass evacuations. These operations can be made by common vehicles such as buses. It can be for example the evacuation of a school or camping that will be impacted by the flood. The vehicles of this category generally have high capacity so we set it to 30. 5 vehicles are considered for the experiments. During the crisis, this category represented 66% of victims rescued through 7 interventions.

Category 2

The interventions gathered in this category need more specific vehicles than the first one. When water already reaches inhabited areas, evacuation is more difficult, and specialized vehicles are needed. These vehicles can go up to 80 centimeters water level in case of emergency but are often limited to 50 centimeters for the safety of the rescue teams. In fact, when the water level is too high, a firefighter scoots in front of the vehicle to detect a potential ditch masked by the water. But this might become dangerous for this scoot in urban areas with sewer drains that might aspirate him for example. These vehicles have limited capacity we set to 10 for 4 vehicles in the experimental fleet. There were 32 demand points of this category during the Luchon crisis for 19% of the victims.

Category 3

When the water level is too high, road vehicles cannot access the area of the intervention. The relief operation is then operated by teams equipped with boats. These boats have a limited capacity set to 5 for the experiments with 3 vehicles of this category covering the crisis. This category affected fewer victims with 8% of them dispatched on 15 nodes.

TABLE 3. Category dispatching summary.

Category Category	Number of nodes	Percentage of nodes	Number of victims	Percentage of victims
1	7	12%	330	66%
2	32	53%	95	19%
3	15	23%	40	8%
4	5	10%	5	1%
5	1	2%	30	6%

Category 4

When none of the resources listed above can rescue a victim, the only way may be to use a helicopter. This kind of resource is very scarce but is available in cases of extreme danger for a victim. We consider only one helicopter available full time for our experiments and its capacity is 1. This category only represents 1% of the victims of the flood with 5 interventions via helicopter for the case of study.

Category 5

This category is specific since it does not consider human victims but animals. In fact, cattle can also be affected by the flooding events and it needs saving too. Usually, the rescue teams act as reinforcement for the cattle's owner and the transportation is operated thanks to the breeder's resources. That is why we only consider one vehicle with a capacity of 10 to relieve this category of victims that represented about 6% of the victims in Luchon on only one node.

To give some perspective to this percentage, it is important to clarify that during the Luchon crisis more than 500 persons or animals were rescued over 60 nodes according to the experience feedback. The data is recapitulated in the table hereunder:

B. GRAPH GENERATION

In order to evaluate the performance of the heuristics presented in this article, the objective, as stated earlier, is not to replay the crisis that happened in 2013 since the data is not complete. The purpose of this generator is to be able to generate a wide variety of graphs depending on the parameters. The process is to use the data extracted from EF in order to generate the demands rescue teams have to intervene. In real-life, rescue teams have access to maps of the impacted area at the beginning of the crisis. These maps display the different stakes of the territory that might turn into demand points at some time of the crisis. This is not exhaustive but gives a good representation of the impacted area. The set of nodes that rescue teams take into account at the beginning of the crisis is a subset of the nodes of this map forming the initial graph. For the purpose of experimentation, the goal is to reproduce the distribution of such nodes over the area of study. In order to do so we developed a graph generator that creates territories. Since as we will discuss later, there might be several different areas of action, we call them sectors. A sector is most of the time an urban area so we model it through concentric circles. The number of zones represented by these concentric circles

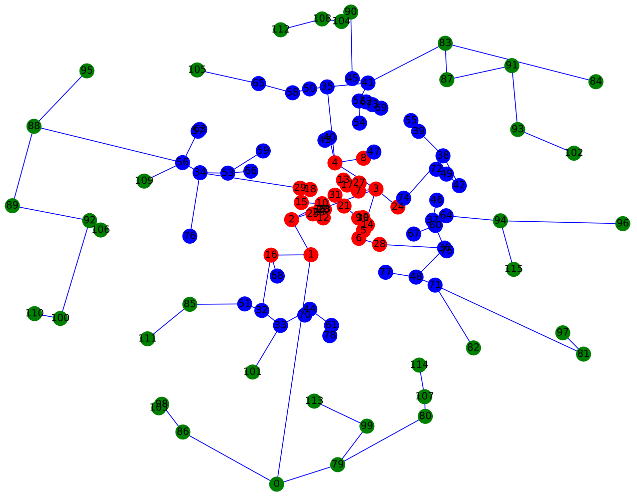


FIGURE 2. Example of generated graph.

TABLE 4. Category dispatching summary.

Zone	Radius	Density	Connectivity
1	1000	10	3
2	2000	5	2
3	4000	1	1

is set at the creation of the sector. These zones represent the different areas of population density from the center of an urban area to the countryside around it. That is why for each zone the population density and the connectivity factor are configurable as well. This last parameter is the number of direct neighbors each node has in the zone. From these factors, nodes are randomly generated on a Cartesian coordinate system as we may observe on Figure2:

We can see the different zones of the same sector differentiated colors on this graph. The numbers inside the nodes just represent the id of the demand. We created it with the following parameters:

Using this generator and a set of parameters chosen to fit the studied area, we generate graph similar to Luchon we call Luchon-like.

At this stage, the graph only represents the potential stakes in a Luchon-like crisis. The next step is to select some of these nodes to become demands for the experiment. In our experiment protocol, we already have the number of demands we desire to try our heuristics on, to simulate a Luchon-like crisis. In order to select the nodes, a selection program was designed to randomly choose the nodes in a specific area of the studied sector. This area is composed of a segmented line representing the riverbed and, for each of these segments, a value is associated to delineate the expansion of the flood from its bed. Values are then associated with these demands for the following variables:

- **Priority**: Uniform distribution among the different priority values. Note that the **deadlines** are associated with the priority.

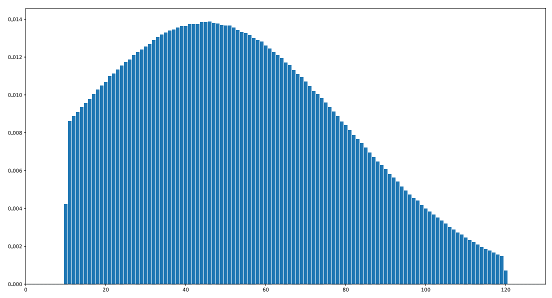


FIGURE 3. Truncated normal distribution with $\mu = 45$, $a = 10$, $b = 120$ and $\sigma = 35$.

TABLE 5. Demands size distribution law by category.

Category	1	2	3	4	5
Distribution Law	Normal $\mu = 45$ $a = 10$ $b = 120$ $\sigma = 35$	Normal $\mu = 3$ $a = 1$ $b = 8$ $\sigma = 2$	Normal $\mu = 3$ $a = 1$ $b = 6$ $\sigma = 2$	Single victims at each node	A single node of 30

- **Action Time**: Uniform distribution on the interval [5, 35]. This interval was given by the rescue teams from SDIS 31, the unity of time is the minute.
- **Category**: Uniform distribution among the 5 categories with a maximum of nodes for each predefined category.
- **Demand size**: The law of distribution for these parameters depends on the category as stated in Table 5.

In this table, we refer to Normal law. In this case, the distribution is truncated so that for all x-value: $a \leq x \leq b$ and the mean of the distribution is equal to μ and the standard deviation is equal to σ.

VI. EXPERIMENTAL RESULTS

The validation of the different algorithms presented in this article is based on Luchon-like graphs. Three experiments are presented in this section. First, we present the performances of the different heuristics. The criterion for this evaluation are both the computation time and objective score (evaluation of the solution proposed by the algorithm through the objective function). The experiment is made on 100 Luchon-like graphs and for each of the three heuristics tested (SDI, BFI, BFIOQ) a mean for the computation time and the objective score is calculated. The results displayed in Figure 4 offer the detail category by category for the Flow-time score in ordinate, the computation time mean is displayed on the abscissa. For this experiment, we also developed the Solomon insertion heuristic from [33] in order to compare our performance to this heuristic from the literature. Note that we did 6 runs with the Solomon heuristic for each graph using only one initialization criteria: the farthest unrouted customer. The second initialization criteria used in the article was not a fit for our model since several demands might have the same deadlines. The parameters used (μ, λ, α_1, α_2) are: (1,1,1,0), (1,1,0,1), (1,1,1,1), (1,2,1,0), (1,2,0,1) and (1,2,1,1). Finally note that the computation time displayed in the results are

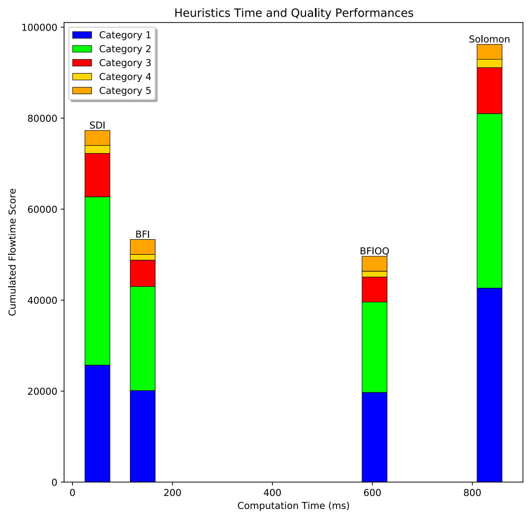


FIGURE 4. Graphic comparing computation time and cumulative objective score performances of the heuristics.

only counting the computation time for the best solution and not for the 6 runs accumulated in order to make the reading of the results easier.

The figures of this graphic are detailed in Table 6:

TABLE 6. Demands size distribution law by category.

	Average OS	Average CT (ms)	Deviation from best OS solution	CT deviation from best OS solution
SDI	77285.22	49.91	55.7 %	-87.8 %
BFI	53329.04	140.49	7.44 %	-78.07 %
BFIOQ	49633.53	604.4	0 %	0 %
Solomon	96211.41	834.45	93.84 %	30.3 %

OS = Objective Score and CT = Computation Time

In detail, these results show a 30% improvement in Objective score from SDI to BFI and 40% from SDI to BFIOQ. These improvements are non-negligible and demonstrate the gain that BFIOQ incurs over the first version of the heuristic BFI. Furthermore, we observe that every developed heuristic shows better performances in terms of computation time and objective than Solomon's heuristic.

However, we can observe that improving the objective score has a cost on the computation time as we see in Figure 4. Note that the categories in the figure are piled up, from one to five from the bottom to the top. SDI has a better computation time than BFI and is almost 15 times faster than BFIOQ. These results are expected since BFIOQ questions the first solution is found and this is time-consuming. Even if this drawback for BFIOQ is not to be totally forgotten, its impact is not as important as its advantages. In fact, the computation time mean for BFI over the 100 graphs is approximately 600 ms and is rarely over the second which fits the requirements for the decision support tool.

This article also tackles the issue of resources dispatch. When the flood impacts several distinctive areas but has to be treated by the same entity, an effective dispatch of resources

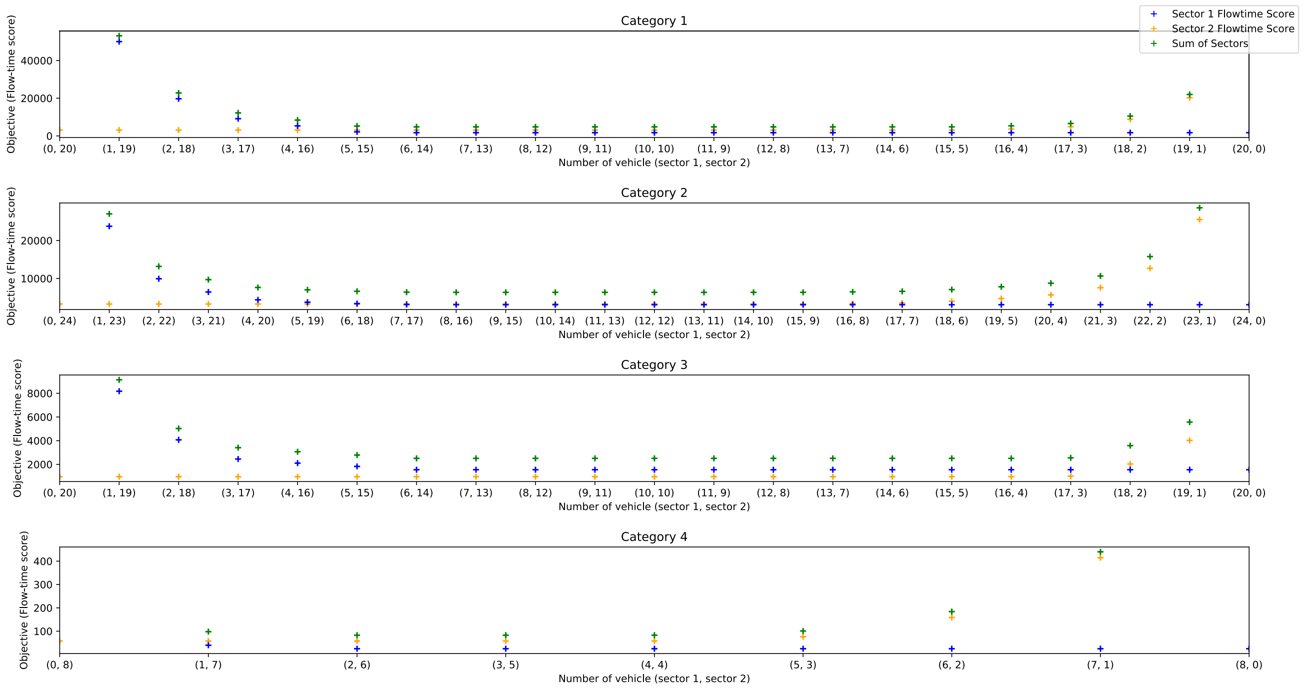


FIGURE 5. Variations of the objective score with affected resources evolution.

through the impacted sectors is essential to optimize the relief operations. Different options studied to optimize this dispatch are presented earlier in this article. We test BFRD on a larger number of vehicles than usual in order to make sure to catch all the variations of the objective with the quantity of resources over the possible value interval. The results of this experiment are presented in Figure 5.

The results are displayed category by category, we choose not to display the 5th category because it is only one demand so its variation is not interesting to study. In the graphics, the number of vehicles allocated to each sector varies on the abscissa (decrease for the first sector and increase for the second one). The sum of both objectives is also displayed in green. These results show that the objective variations are constant, the objective always decreases with the increase of the resources or is constant. These results justify the development of an improved version of BFRD we also presented: CODRD.

As for the previous experiment, the resources dispatch algorithm is tested on 100 graphs, and each of them is split into two sectors. In fact for this experiment, 100 Luchon-like graphs have been generated and then split into two sectors where resources are dispatched. The performance results are the mean of the objective score over these graphs. However, as mentioned in Section IV-E, the greedy algorithm might induce some infeasible situation. We then mean only the graphs that are feasible by all 5 algorithms. The results are displayed in Figure 6 where the percentage of infeasibility as defined earlier is written over the bars of each algorithm. Note that the categories in the figure are dispatched from one to five from the bottom to the top. The experiments were built

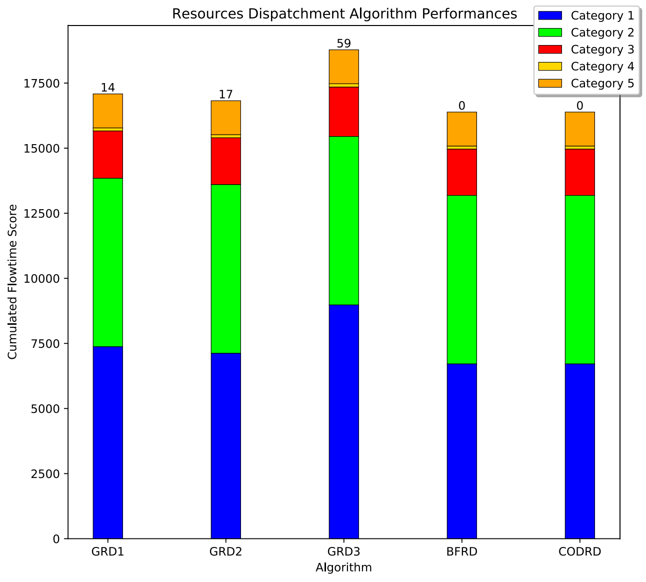


FIGURE 6. Objective Score performances for the different resources dispatch algorithms.

using BFIOQ as a heuristic since it is the heuristic showing the best objective score performances.

In this figure, we observe that GRD3 implies a lot of infeasible situations. The choice has been made to keep it in the displayed results even if it reduces the number of graphs on which the mean is computed. The reason is that we tried the experiment without it and the performance results are slightly the same. We observe as expected that in general, greedy algorithms induce a non-negligible infeasibility ratio which is not the case for the other algorithms. Furthermore, the performances of CODRD are 4% better than those of GRD1 which is the best greedy algorithm. However, the computation time in this situation is not negligible contrary to the previous experiment and it needs to be taken into account. The average computation time for the greedy algorithms is the same as those of the heuristic used (BFIOQ) so under the second. The results show that CODRD needs 6 runs instead of 1 for the greedy algorithms on average and the computation time follows the same logic with a computation time 6 times superior. We consider that in a situation where the resources sufficient enough to avoid to be a hard constraint on the problem, the use of a greedy algorithm such as GRD1 is probably the most appropriate choice regarding the few performance gain from the use of CODRD compared to the computation time gain. On the opposite, we prefer to use CODRD in situations where resources are rare and the computation time is worth the gain because at list a solution is guaranteed where GRD1 shows a 14% error rate (dispatching problems where the greedy algorithms dispatch led to an infeasible situation for the heuristic BFIOQ).

Despite the good performances of the heuristics observed on the experimental set, it might be interesting to get some perspective on the reaction of the heuristics to scenarios of different nature. Some characteristics of the data-set could lead to different performances from the heuristics as follows:

- The number of nodes in the graph is a key factor for the computation time performances. In fact, dealing with an NP-complete problem, we expect to have exponential computation time with the size of the problem as studied in [18] using MIQP. This problem is also expected to arise at a lower scale with the heuristics, most specifically with BFIOQ that includes an optimization on turns that tries all the orders possible. It would be interesting for further studies to take good care of experimenting with larger-scale scenarios.
- The distribution of the quantity among demands might also have an impact on performances. For example for a total number of 20 victims dispatched on 4 nodes, the scenarios where there are 5 victims at each node or 1 victim at 3 nodes and the 17 other victims at one node will most likely require different routes to follow in order to optimize the response. In the first case the travel to a node will have the same benefit on the objective for every node whereas in the second one we still have to travel to all the nodes but of them will only be to rescue one person. The impact on the objective of each node of the graph is less balanced in this second case. This could be interesting to create an index to measure the distribution of victims among nodes and experiment with different distributions instead of mimicking Luchon's victim distribution.
- In this paper we worked with 4 levels of priorities but this scale could be smoothed on a larger range of values in order to improve the treatment of the demands. It would probably not show better performances in terms of objective since the priority is one of the factors taken into the computation of the objective score but it might induce better specific handling of each demand.
- The nature of the graph itself is a major issue. Depending on whether we deal with high distances and few connectivity or short distances and high connectivity, the constraints for the heuristics are different. These limit cases could represent respectively a crisis in a countryside territory with few inhabitants compared to flooding in a big city. This problem can be solved by the rescue teams with the dimensioning of the vehicle fleet and clustering. The same solution might be applied in the case of dealing with very high action time values.

Finally this article focus on developing algorithms sized to answer to real data and problem characteristics.

VII. CONCLUSION

In this article, we developed the problem encountered when trying to optimize the rescue teams' response to flashflood. After exposing the complete model we demonstrated that the use of heuristics is the best fit for the computation time requirements for a real-time support decision tool. The heuristics BFI and its improved version BFIOQ were presented and compared to the heuristic SDI that is meant to reproduce the current behavior of the rescue teams. The comparison made on a large set of graphs built to reproduce conditions from the Luchon flooding from 2013, shows that the use of BFIOQ helps improve the objective performance by more than 55% with an affordable cost on computation time. In fact, SDI computation is completed under 100ms and BFIOQ over 600ms which makes SDI 87% faster on average, the computation time for BFIOQ stays under the second which is satisfying. Furthermore, we compared BFIOQ to a widespread algorithm from the VRP's literature, Solomon insertion Heuristic. The results show that this algorithm does worth on both comparison criteria with almost 2 times worth performances in objective and a 30% increase of computation time. These results might be explained by the choice of the objective that this heuristic was not developed to respond to.

Several algorithms of different types were also developed to tackle the issue of resources dispatch over several impacted sectors. Their comparison on the same kind of graphs mentioned earlier but adapted to two sectors shows that greedy algorithms like GRD1 allow good performances in a very short amount of time but can create infeasible situations while heuristics like CODRD allow to make sure not to create an infeasible situation and are slightly more efficient in objective score aspect (5% approximately) but with a huge cost on computation time (computation time 6 times higher for CODRD). This raises the question of its usage in a final tool where the common time expensive operations could be gathered to speed up the process. Also, the importance of the computation time varies during the different phases of the crisis, this can be made into perspectives accordingly to the approach chosen to tackle the problem dynamically.

If we choose a re-optimization approach the computation time is a central issue however if another approach is preferred such as insertion heuristics the computation time is less of an issue.

REFERENCES

[1] M. Allahviranloo, J. Y. J. Chow, and W. W. Recker, "Selective vehicle routing problems under uncertainty without recourse," *Transp. Res. E, Logistics Transp. Rev.*, vol. 62, pp. 68–88, Feb. 2014.

[2] E. Angelelli, N. Bianchessi, R. Mansini, and M. G. Speranza, "Short term strategies for a dynamic multi-period routing problem," *Transp. Res. C, Emerg. Technol.*, vol. 17, no. 2, pp. 106–119, Apr. 2009.

[3] C. Archetti, M. G. Speranza, and A. Hertz, "A Tabu search algorithm for the split delivery vehicle routing problem," *Transp. Sci.*, vol. 40, no. 1, pp. 64–73, Feb. 2006.

[4] C. Archetti, D. Feillet, A. Mor, and M. G. Speranza, "Dynamic traveling salesman problem with stochastic release dates," *Eur. J. Oper. Res.*, vol. 280, no. 3, p. 832–844, 2020.

[5] C. Archetti, E. Fernández, and D. L. Huerta-Muñoz, "The flexible periodic vehicle routing problem," *Comput., Oper. Res.*, vol. 85, pp. 58–70, Sep. 2017.

[6] C. Archetti and M. G. Speranza, "The split delivery vehicle routing problem: A survey," in *The Vehicle Routing Problem: Latest Advances and New Challenges*, vol. 43, B. Golden, S. Raghavan, and E. Wasil, Eds. Boston, MA, USA: Springer, 2008, pp. 103–122.

[7] F. Baniel, "Prise en compte d'objectifs de stabilité pour l'organisation de collectes de déchets," M.S. thesis, Thierry et Huguet, Marie-José Systèmes Industriels, Toulouse, France, 2009.

[8] R. Bent and P. Van Hentenryck, "A two-stage hybrid local search for the vehicle routing problem with time windows," *Transp. Sci.*, vol. 38, no. 4, pp. 515–530, Nov. 2004.

[9] J. Berger, M. Barkaoui, and O. Bräysy, "A route-directed hybrid genetic approach for the vehicle routing problem with time windows," *Inf. Syst. Oper. Res.*, vol. 41, no. 2, pp. 179–194, May 2003.

[10] D. Berkoune, J. Renaud, M. Rekik, and A. Ruiz, "Transportation in disaster response operations," *Socio-Economic Planning Sci.*, vol. 46, no. 1, pp. 23–32, Mar. 2012.

[11] N. Bianchessi and G. Righini, "Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery," *Comput. Oper. Res.*, vol. 34, no. 2, pp. 578–594, 2007.

[12] M. J. Booij, "Impact of climate change on river flooding assessed with different spatial model resolutions," *J. Hydrol.*, vol. 303, nos. 1–4, pp. 176–198, 2005.

[13] T. Chabot, F. Bouchard, A. Legault-Michaud, J. Renaud, and L. C. Coelho, "Service level, cost and environmental optimization of collaborative transportation," *Transp. Res. E, Logistics Transp. Rev.*, vol. 110, pp. 1–14, Feb. 2018.

[14] L. Chen and E. Miller-Hooks, "Optimal team deployment in urban search and rescue," *Transp. Res. B, Methodol.*, vol. 46, no. 8, pp. 984–999, 2012.

[15] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manage. Sci.*, vol. 6, no. 1, pp. 80–91, 1959.

[16] M. Dror and P. Trudeau, "Savings by split delivery routing," *Transp. Sci.*, vol. 23, no. 2, pp. 141–145, 1989.

[17] M. Dror and P. Trudeau, "Split delivery routing," *Nav. Res. Logistics*, vol. 37, no. 3, pp. 383–402, 1990.

[18] F. Dubois, P. Renaud-Goud, and P. Stolf, "Capacitated vehicle routing problem under deadlines," in *Proc. Int. Conf. Inf. Commun. Technol. Disaster Manage. (ICT-DM)*, Dec. 2019, pp. 1–8.

[19] M. R. Garey and D. S. Johnson, "*Computers and Intractability*, vol. 29. San Francisco, CA, USA: Freeman, 2002.

[20] M. Gendreau, G. Laporte, and R. Séguin, "An exact algorithm for the vehicle routing problem with stochastic demands and customers," *Transp. Sci.*, vol. 29, no. 2, pp. 143–155, May 1995.

[21] B. Golden and A. Assad, "Vehicle routing with time-window constraints," *Amer. J. Math. Manage. Sci.*, vol. 6, nos. 3–4, pp. 251–260, 1986.

[22] B. Golden, A. Assas, L. Levy, and F. Gheysens, "The fleet size and mix vehicle routing problem," *Comput. Oper. Res.*, vol. 11, no. 1, pp. 49–66, 1984.

[23] J. Homberger and H. Gehring, "A two-phase hybrid metaheuristic for the vehicle routing problem with time windows," *Eur. J. Oper. Res.*, vol. 162, no. 1, pp. 220–238, Apr. 2005.

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

[25] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 3, pp. 345–358, 1992.

[26] H. Larrain, L. C. Coelho, C. Archetti, and M. G. Speranza, "Exact solution methods for the multi-period vehicle routing problem with due dates," *Comput. Oper. Res.*, vol. 110, pp. 148–158, Oct. 2019.

[27] A. Larsen, "The dynamic vehicle routing problem," M.S. thesis, Dept. IMM, Tech. Univ. Denmark, Lyngby, Denmark, 2000.

[28] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Comput. Oper. Res.*, vol. 34, no. 8, pp. 2403–2435, Aug. 2007.

[29] D. Reyes, A. L. Erera, and M. W. P. Savelsbergh, "Complexity of routing problems with release dates and deadlines," *Eur. J. Oper. Res.*, vol. 266, no. 1, pp. 29–34, Apr. 2018.

[30] I. Rodríguez-Martín, J.-J. Salazar-González, and H. Yaman, "The periodic vehicle routing problem with driver consistency," *Eur. J. Oper. Res.*, vol. 273, no. 2, pp. 575–584, Mar. 2019.

[31] N. Secomandi and F. Margot, "Reoptimization approaches for the vehicle-routing problem with stochastic demands," *Oper. Res.*, vol. 57, no. 1, pp. 214–230, 2009.

[32] J.-B. Sheu, "An emergency logistics distribution approach for quick response to urgent relief demand in disasters," *Transp. Res. E, Logistics Transp. Rev.*, vol. 43, no. 6, pp. 687–709, Nov. 2007.

[33] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, 1987.

[34] Y. Tramblay and S. Somot, "Future evolution of extreme precipitation in the Mediterranean," *Climatic Change*, vol. 151, no. 2, pp. 289–302, Nov. 2018.

[35] F. Vinet, D. Lumbroso, S. Defossez, and L. Boissier, "A comparative analysis of the loss of life during two recent floods in France: The sea surge caused by the storm Xynthia and the flash flood in Var," *Natural Hazards, J. Int. Soc. Prevention Mitigation Natural Hazards*, vol. 61, no. 3, pp. 1179–1201, Apr. 2012.

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

[37] K. G. Zografos and K. N. Androutsopoulos, "A decision support system for integrated hazardous materials routing and emergency response decisions," *Transp. Res. C, Emerg. Technol.*, vol. 16, no. 6, pp. 684–703, 2008.