# Dynamic Capacitated Vehicle Routing Problem for Flash Flood Victim's Relief Operations

**Florent Dubois**
Institut de Recherche en Informatique de Toulouse – Université de Toulouse
florent.dubois@irit.fr

**Paul Renaud-Goud**
Institut de Recherche en Informatique de Toulouse – Université de Toulouse
paul.renaud.gouds@irit.fr

**Patricia Stolf**
Institut de Recherche en Informatique de Toulouse – Université de Toulouse
patricia.stolf@irit.fr

## ABSTRACT

Flooding relief operations are Dynamic Vehicle Routing Problems (DVRPs). The problem of people evacuation is addressed and formalized in this paper. Characteristics of this DVRP problem applied to the crisis management context and to the requirements of the rescue teams are explained. In this paper, several heuristics are developed and assessed in terms of performance. Two heuristics are presented and adapted to the dynamic problem in a re-optimization approach. An insertion heuristic that inserts demands in the existing plan is also proposed. The evaluation is conducted on various dynamic scenarios with characteristics based on a study case. It reveals better performances for the heuristics with a re-optimization approach.

## Keywords

Vehicle Routing Problem, Flash floods, Dynamic

## INTRODUCTION

In recent years, research has been conducted on the relations between climate change and the increase of disastrous events especially flooding events as studied in Booij 2005 and Tramblay and Somot 2 018. These events have a high impact on people and infrastructures. In fact Vinet et al. 2012 offers a summary of the human casualties and infrastructures damages from flooding events in France from 1983 to 2010 and estimates at least 252 human lives losses and 8,3 billion euros worth of damages. The ANR e-Flooding project Stolf et al. 2019 tackles this issue and aims at reducing the impact of flooding e vents. In collaboration with the SDIS 31 (firefighter entities for the Haute-Garonne department in the South of France in charge of rescue operations in flooding context) in this research project, this article focuses on the short-term response to flooding and more precisely flash floods. Flash flood is a type of flood where the water level rises in a very short time period leaving no time for rescue teams for anticipation. The faced problem considers people relief operations where victims need to be picked up by rescue vehicles. The victims are then conducted to the depot (rescue center) and considered safe once they are in the vehicles. These decision problems are known in the literature as Vehicle Routing Problem (VRP) and try to improve the routing of one or several vehicles in order to serve multiple locations.

One of the challenges in this problem is to plan the interventions for all the vehicles available for the rescue teams and optimize them. More specifically in this problem, vehicles have different specialties for interventions called categories. Furthermore, the capacity limitations of the vehicles have to be considered, which characterizes our problem as a Capacitated Vehicle Routing Problem (CVRP). In addition, some of the demands have too many victims to be rescued by a single-vehicle so the demands will have to be split. Moreover, resources do not allow to rescue all victims with one tour starting from the depot and back to it with relieved victims. Therefore plans have to be made on several tours in order to empty vehicles and serve new locations. The emergency aspect of the problem has also to be taken into account, rescue teams indeed associate to each demand a priority based on several factors such as the vulnerability of victims. This priority is also associated with a deadline: the worst-case service date. This classifies our problem also as a VRP with Time Windows introduced in Golden and Assad 1986. The demand needs to be served after the beginning of the time window which here is the release time of the demand and before the end of the time window: the deadline. A set of demands is known at the beginning of the crisis, but most of the demands are revealed during the crisis for example through rescue team's scouting or calls from victims. These events are dynamic and need to be served therefore the plans have to be updated dynamically. This characterizes this problem as a DVRP. The different types of considered dynamic events and the way to deal with them are detailed later in this paper.

A wide literature exists on the DVRP but mainly on commercial issues. These problems do not consider priorities on requests and often treat deadlines as relaxed constraints applying a penalty in case of violation. Furthermore, computation time is not as important in such applications since they have a preparation period to compute initial routes, and dynamic events are less frequent or less urgent to handle. The Crisis Management context, and more specifically, the flash flood context, on the opposite, does not leave preparation time for rescue teams. Dynamic events might be emergency requests that need to be handled as soon as possible. A support decision tool for this type of problem must be able to answer such requirements and offer a solution to rescue teams in a short amount of time for emergencies. Other Crisis Management problems often consider the delivery of supply to impacted areas or the optimal deployment of a fleet of rescue teams, but none of them, at our knowledge, considers victims relief operations as a VRP using a priority factor as the one used by rescue teams and central in our model.

This article proposes heuristics to this dynamic problem in order to answer the need of rescue teams. Strategies are presented that define when re-optimizations are computed in relation to dynamic events releases. Several algorithms are also presented and evaluated. An evaluation process with dynamic scenarios is conducted to assess heuristics performances. This evaluation uses a simulator developed for this study that runs scenarios in simulated time and models the communication of the vehicles with the Decision Center (DC) using a multi-process environment. Coupled with a graph generator that allows the creation of customized crisis scenarios with adapted metrics that are presented in this article, this simulator enables observation of the solutions' performances. The contributions of this article are listed below:

- The model of the problem and the mechanisms to handle dynamic events
- Several dynamic heuristics and the comparison of their performances on various types of scenarios
- An integration of a measure of dynamism and the definition of a demand distribution index in order to control temporal and quantity aspect of the crisis scenarios
- A graph generator that generates scenarios with controlled values of dynamism and distribution
- A simulator to play crisis scenarios in fictive time and synchronize processes to recreate communication between vehicles and the DC

The remainder of this paper is organized as follows. Section State of the art presents the literature around the studied problem with a focus on the DVRP. Section Dynamic Problem describes the problem and explains the mechanism to handle a dynamic problem. The dynamic scenarios and the tools to run them are presented in section Dynamic Scenarios. The section Solutions presents the solutions developed to handle the problem. Finally the experimental results are presented in section Experimental Results before Conclusion.

## STATE OF THE ART

The VRP problems have been studied through various problems with very diverse characteristics as listed in Laporte 1992. The category of VRP that the article focuses on is the DVRPs. It is characterized by the reveal of information or events during the execution of the previous plans during the crisis and over the time horizon. These events might be of diverse types. The most widespread sources of dynamism in the literature according to Ritzinger et al. 2016 are:

- Dynamic Customers: It describes problems where the points of demands (location and characteristics) are revealed dynamically. For example Gendreau, Laporte, et al. 2001 studies the relocation and deployment of ambulances with interventions unknown in advance.

- Dynamic Requests: In opposition to dynamic customers, information about the demand is known but the quantity of goods or persons to pick up or deliver at the location is revealed during the problem. Sometimes not before arrival at the node. This is the case of Van Hentenryck et al. 2010 where information is confirmed upon arrival. They however have probabilistic information about the requests that help define online routing strategies.
- Dynamic Travel Times: In these dynamic problems, the travel times of the vehicles are considered to be variable over time like in Vu et al. 2020. This article studies the Traveling Salesman Problem (TSP) where the travel time can vary with the date of departure.

Our problem gathers these three sources of dynamism. Indeed it is considered that the travel time might vary during the crisis because of the crisis management context and its unpredictable events such as a cut road. The problem can also be considered as a dynamic customer problem with the release of new demands over the crisis. For example with victims that call the emergency services to ask for an evacuation. Finally, our requests are also dynamic. Information might vary like the number of victims, the priority of the node, and the service time (time necessary to complete the relief operations at the node). DVRPs are divided into different types of problems as confirmed by Pillac et al. 2013. Stochastic VRPs or purely Dynamic VRPs.

Uncertainty on Stochastic VRPs can be treated over worst-case scenario as in Ilgaz et al. 2008. This approach is used to solve a stochastic requests problem with an exact method. R. Bent and Van Hentenryck 2003 and R. W. Bent and Van Hentenryck 2004 use a multiple-plan generation approach to solve stochastic customers problems by continuously generating plans for every possible value of the uncertain variables. This approach is suited for problems with a limited range of uncertain variable values. W.-H. Yang et al. 2000 used the same approach to generate multiple routes minimizing route costs on stochastic customer and request problems. The solution chosen is a restocking strategy where the purpose is to decide the best moment to go back to the depot for vehicle refilling. Policies can also be settled to treat dynamic information as in Van Hentenryck et al. 2010 or Albareda-Sambola et al. 2014 to estimate the best time period to serve stochastic requests. Archetti et al. 2020 uses a re-optimization approach, often used in purely dynamic problems, coupled with a Markov Decision Process to deal with stochastic release dates of customers for a parcel delivery problem.

Necessarily when no information is known in advance in DVRPs, other strategies have to be used. A lot of dynamic approaches use re-optimization, which means a static algorithm is run periodically or continuously to update the solution. Gendreau, Guertin, et al. 2006 uses this approach with a tabu search and a neighborhood heuristic. The algorithm is run at every new dynamic event.

When the dynamic of the crisis allows it, the exact method can better solve the problem, they often give better solutions even if they need more time to compute, like in Vu et al. 2020. But the re-optimization might be periodical as experienced in Archetti et al. 2020 that also used a deterministic model and tried to vary the period of re-optimization or J. Yang et al. 2004 that combines exact method and re-optimization policies. Policies are rules that state how to deal with each type of dynamic event depending on certain characteristics.

However, when the computation time is a strong constraint of the problem, routing policies are often used. A routing policy states step by step which demand is to be served next without establishing a plan for several future interventions. Potvin et al. 2006 tested different routing policies relevant to several cases including emergency services. Angelelli et al. 2009 also studied this category of solutions.

Hybrid options have also been studied like multiple horizon approach that uses strategies for a short term horizon and another approach for a long term horizon as Mitrović-Minić et al. 2004 that uses an improvement heuristic on a long term horizon. Wen et al. 2010 even combined three approaches in a three-phase rolling horizon strategy. Finally Larrain et al. 2019 use local search embedded in a tabu search in order to find better solutions. This last approach is interesting for the problem studied in their paper but the tabu search demands too much computation time and our paper does not deal with a problem over several days as in Larrain et al. 2019 but only a few hours.

In light of the solutions offered in the literature and regarding the specificity of our problem, this paper offers different solutions to the Dynamic Capacitated Vehicle Routing Problem under Deadlines (DCVRPD) that fit the particular context and dynamic of a crisis management problem. It helps to answer the rescue teams' (as SDIS 31) specific requirements and needs. These solutions are evaluated on different metrics for solutions quality as well as computation time which is a strong constraint in our problem. A wide literature exists on the DVRP and a part of it focuses on crisis management problems, however, to the best of our knowledge, there is no work dealing with rescue operations as a VRP that gathers all the constraints of our problem. Due to the specific context of flash floods, most solutions are not adopted because of their computation time, too important to be adapted. That is why we believe this article is a significant contribution to the domain.

## DYNAMIC PROBLEM

### Problem description

A crisis scenario is described as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the vertex set $\mathcal{V} = [0, V]$ of size $V + 1, V \in \mathbb{N}$. The vertices of this graph are the demand points of the problem and the edges represent the shortest path between locations. The vertex 0 is the rescue center where victims are driven by rescue teams and considered in safety. $\mathcal{V}^\star$ is the subset of $\mathcal{V}$ which excludes the rescue center. Edges are associated with travel time: for all $i$ and $j$, $tt_{i,j,c} \forall (i, j) \in \mathcal{V}^2$ with $c$ the category of the vehicle. Different categories of vehicles are considered in this paper. Both nodes and vehicles are associated with a category.
Nodes $i \in \mathcal{V}$ of the graph are described by:

- A demand $d_i$: number of victims
- A service time $a_i$: time to rescue the victims
- A category $c_i$
- A priority $p_i$: value of the priority factor determined by the rescue teams
- A deadline $f_i$: the latest date to serve the demand
- A release time $r_i$ date at which the demand is known

The objective is to optimize vehicles plans for multiple vehicles. Let $\mathcal{M}$ be the set of vehicles. To each $k \in \mathcal{M}$ is associated a category $cat_k$ corresponding to the category of nodes the vehicle can serve. Vehicles also have a maximum capacity noted $Q_k$. Since the fleet of vehicles is rarely enough to serve all demands in one tour because of capacity limitations, plans are made on several tours $z \in \mathcal{Z} = [1, Z]$ where $Z$ is the maximum number of tours, separated by a passage through the rescue center to empty the vehicle.

In order to fully qualify a solution to the problem, 3 variables are needed:

- $q^z_{i,k}$: number of victims to rescue by vehicle $k$ at node $i$ on tour $z$
- $h^z_{i,k}$: date of arrival of vehicle $k$ at node $i$ on tour $z$
- $x^z_{i,j,k}$: Boolean equals to 1 if vehicle $k$ travels from node $i$ to node $j$ on tour $z$

Here follows the objective function:

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

The objective function (1) is based on the notion of Flow-time which is the time between release and service of a demand. It is the sum of Flow-times $h^z_{i,k} - r_i$ for each intervention of rescue vehicles, weighted by the priority factor of the demand and the number of persons rescued at node $i$ by vehicle $k$ on tour $z$.

### Dynamic Mechanisms

In real-life situations, there exist some demands that are not known at the beginning of the crisis and dynamic events occur. In this work, different types of events are considered:

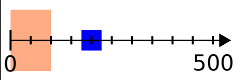
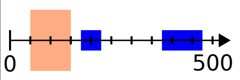
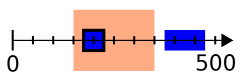
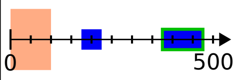
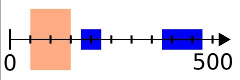
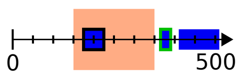
- New demands: Information about a new demand to be served is released through scouting or calls. This creates a new request: a node is added to the graph and needs to be handled in the next plan update.
- Delays: When a vehicle is late regarding its plan after travel or service on a point of demand, it produces a delay. This event allows to update the knowledge of the situation in the decision center and may update the plans accordingly.

Solving a dynamic problem raises several challenges. On one hand, there is a competition between the vehicles serving their missions, and the decision center that tries to improve the plan. The decision center needs time to compute and assign routes and missions to vehicles, but the vehicles are in motion. Hence new missions should not be set in the near future. On the other hand, a decision needs to be made about when to relaunch computation to update the plan.

In other words, according to the new events that are not yet taken into account, the question is: when is it worth recomputing a new plan?

When we start a plan computation, a neutralized period called the Re-Computation Period (RCP) is defined from the current date. The routing included in this period is immune to changes. In the case where computation is still running when the end of this period is reached, the length of this neutralized period is doubled and the plan computation is relaunched. This event is called an overtime. In fact, when this period is over, the decision center could be questioning missions that are being served in the meantime, making the new plan obsolete.

The initial value for RCP is determined by experimentation. Figure 1 presents an example where the principle of the RCP is illustrated on a single vehicle with a re-computation of the plans that might have been triggered by different types of events. The RCP is illustrated through the second computation, interrupted because the computation time reached the RCP value and it is not guaranteed that we would not question the demands that are being served. Also, the third computation where the RCP is increased after interruption of the previous computation, shows missions that are not questioned because in the RCP, and a mission later in the plan that is offset.

| Re-Computation number | 1 | 2 | 3 |
|---|---|---|---|
| Date | 0 | 50 | 150 |
| Re-Computation Period | 100 | 100 | 100 x2 = 200 |
| Computation Time | 50 | Overtime Interrupted | 150 |
| Vehicle Plan Before Re-Computation |  |  |  |
| Vehicle Plan After Re-Computation |  |  |  |

🟦 Editable Mission 🟩 New Mission ⬛ Fixed Mission 🟧 Re-Computation Period

**Figure 1. RCP example (dates and computation times are expressed in seconds)**

A second mechanism is necessary to decide when to start a new computation. Three strategies have been proposed. This first mechanism RCP is common to all three strategies.

- **Preemptive**: Computation is re-launched at every dynamic event. When a new event is released, computation is interrupted and re-launched with the computation of the interrupted events incremented with the new one.
- **Continuous**: Computation is re-launched as soon as the previous computation ended, with the events that were released during the previous computation but only after this computation is over. If no event was released, the next event triggers the computation of a new solution.
- **Buffered**: Computation is launched at **RCP** interval with all the dynamic events since the previous computation. Contrary to the continuous strategy, once the computation is over, it waits for the end of the RCP period and the events released during this period are buffered.

Figure 2 illustrates how the different strategies work showing when computation of updated solutions is started depending on the release of new dynamic events. In this figure, the overtime process is not illustrated to clarify the explanations, but note that it does not affect how these strategies work. The buffers that have been displayed show

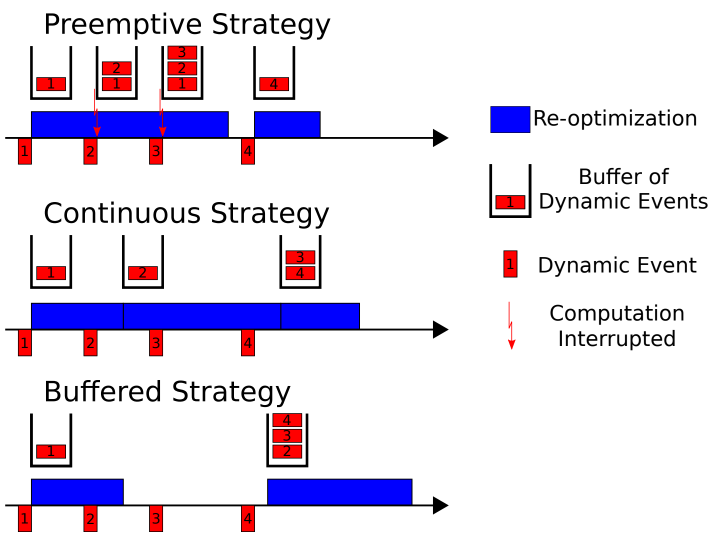


**Figure 2. Dynamic Strategies**

the events that are taken into account in the computation under it. Therefore when an event disappears from the buffer, this means it has been integrated into the plan.

To evaluate a dynamic problem, dynamic scenarios are generated. A dynamic scenario is constituted of data about dynamic events and data of the graph representing the impacted area.

## DYNAMIC SCENARIOS

To evaluate the characteristics of dynamic scenarios, metrics are defined. The dynamism used in the literature represents the temporal distribution of the dynamic events over the time horizon of the crisis. The time horizon is the period in which the problem is studied. For a dynamic problem, it represents the period from the start of the crisis to the last dynamic event release. The dynamism measure is detailed in this section. Another characteristic of the dynamic requests is the number of victims on each node called *quantity* to fit VRP literature vocabulary. A new metric is defined in this section to evaluate the distribution of the number of victims among all the nodes. It is called Quantity Distribution Index (QDI) and is coupled with dynamism metric to generate data for dynamic scenarios.

### Metrics

#### *Measure of Dynamism*

To measure dynamism, different metrics are used in the literature. REF LARSEN represent time before the deadline tor rescue, which could be called the urgency. In our problem, dynamism represents time between demand releases.

In our problem, the deadline means value is bigger than the average time between two releases. Therefore, the main challenge is to handle many demands simultaneously and not to handle demands with short deadlines but spaced on the time horizon. It is important to translate the temporal distribution of dynamic requests releases. As the most appropriate to measure dynamism in our problem we choose the measure defined in VanLon et al. 2016

It suggests an approach that uses a 100% dynamic case as a reference for dynamism. This case is considered to be the most dynamic case possible given a scenario: release times for demands are equally distributed over the time horizon. In our problem, the deadline means value is bigger than the average time between two releases. In the rest of the article, the word "dynamism" refers to the measure defined in VanLon et al. 2016. First the inter-arrival times $\Delta$ is:

$$\Delta := \{\delta_1, ..., \delta_{|\mathcal{V}^\star|-1}\} = \{r_j - r_i, j = i - 1 \quad \forall i, j \in \mathcal{V}^\star\} \tag{2}$$

Then the 100% case is defined as the case where $\forall i \in \mathcal{V}^\star, \delta_i = \theta$ with $\theta = \frac{T}{|\mathcal{V}^\star|}$ the perfect inter-arrival time as shown in fig. 3 . From this reference case, VanLon et al. 2016 defines a deviation between the release time of the demand and the one it would have in the 100% case, for every demand in $\mathcal{V}^\star$:

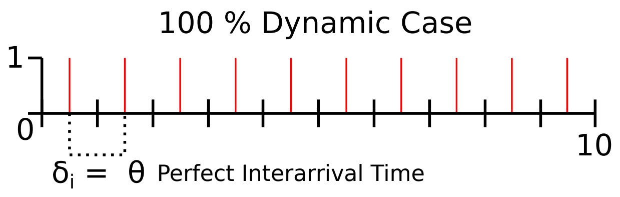


**Figure 3. Example of the** 100% **case**

$$\sigma_i = \begin{cases} \theta - \delta_i & \text{if} \quad i = 1 \quad \text{and} \quad \delta_i < \theta \\ \theta - \delta_i + \frac{\theta - \delta_i}{\theta} \cdot \sigma_{i-1} & \text{if} \quad i > 1 \quad \text{and} \quad \delta_i < \theta \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Consequently the total deviation on the entire scenario is defined as:

$$deviation := \sum_{i=1}^{|\mathcal{V}^\star|} \sigma_i \tag{4}$$

In order to normalize the measure of dynamism, VanLon et al. 2016 also defines the maximum deviation:

$$max\_deviation := \sum_{i=1}^{|\mathcal{V}^\star|} \overline{\sigma_i}, \text{ where} \tag{5}$$

$$\overline{\sigma_i} = \theta + \begin{cases} \frac{\theta - \delta_i}{\theta} \cdot \sigma_{i-1} & \text{if} \quad i > 1 \quad \text{and} \quad \delta_i < \theta \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

And finally:

$$Dynamism = 1 - \frac{deviation}{max\_deviation} = 1 - \frac{\sum_{i=1}^{|\Delta|} \sigma_i}{\sum_{i=1}^{|\Delta|} \overline{\sigma_i}} \tag{7}$$

Another metric is defined to translate the capacitated aspect and the balance of the quantity among demands of the CVRP.

### *Measure of Quantity Distribution*

The process uses the same idea as dynamism using a 100% case as a reference. This metric is called the QDI. It refers to the 100% Distributed Case, which is when the quantity of every demands is equal to the same Perfect Quantity Distribution (PQD) defined $\beta$ as:

$$\forall i \in \mathcal{V}^\star \quad q_i = \beta = \frac{\sum_{i=1}^{|\mathcal{V}^\star|} q_i}{|\mathcal{V}^\star|} \tag{8}$$

In this work, the quantity is distributed by setting the total number of victims according to the problem characteristics and then dispatch them among the nodes. This case is only reachable when the total quantity among demands is divisible by the number of demands which implies that some scenarios will not have a 100% Distributed Case. A deviation from the 100% Case is defined: $\rho_i, \forall i \in \mathcal{V}^\star$. It is the difference between the quantity at node i and the PQD $\beta$:

$$\rho_i = |\beta - q_i| \quad \forall i \in \mathcal{V}^\star \tag{9}$$

The maximal deviation is defined considering that the minimum quantity for a demand is 1. The maximum quantity is reached when all nodes have a minimum quantity except 1 node that hosts the rest of the victims.

To conclude, the scenarios for experimentation are characterized using the following two metrics:

- the dynamism from VanLon et al. 2016 to measure the temporal dynamics of the crisis in terms of distribution of dynamic demands release times,
- the QDI to measure the distribution of quantity among demands.

## Graph and Data Generation

Solutions have been developed to answer a specific problem with a case study as a reference. Benchmarks from the literature do not contain all information necessary for evaluation such as priority. Furthermore, we want to conduct experimentation on instances similar to the case study of Luchon 2013. That is why, to evaluate solutions, an experimental set is built. For this purpose, a graph generator has been developed that allows the generation of parameterized graphs. The generator is based on zones to model the different areas of people density on the field. A zone is associated with a size, a density of nodes, and a degree, which is the number of other nodes each node is directly connected to. This zone has been represented as circles centered on the same point. Figure 4 displays an example of a graph built with 3 zones (represented by 3 colors). The first zone has a degree of 3 and a density of 10 nodes per $km^2$, the second a degree of 2 and a density of 5 nodes per $km^2$, and the third a degree of 1 and a density of 2 nodes per $km^2$. These zones are not of the same sizes. The first has a radius of 1000m, the second 2000m, and the last one of 4000m.

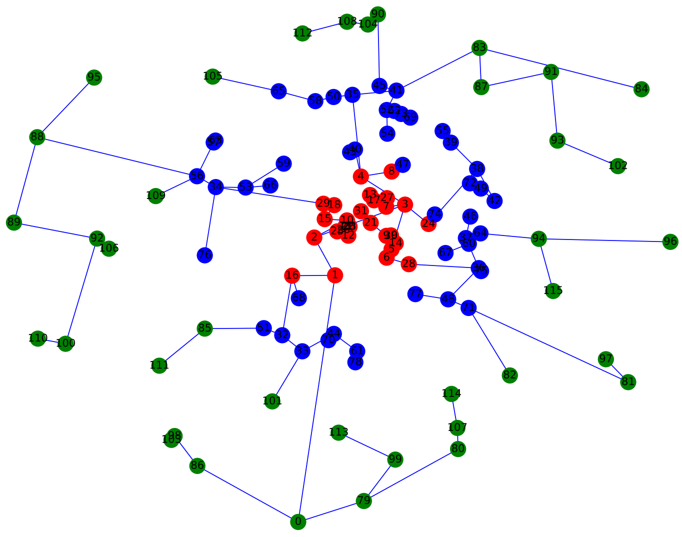


**Figure 4. Example of generated graph**

This configurable graph generator allows creating graphs modeling diverse types of territories from a very populated city where the central zones might be set to a high degree and density, to rural areas with few inhabitants and therefore fewer victims and also few roads, implying link issues. Furthermore, once the graph has been built, the nodes that turn into demand are selected along an impacted area represented through a polygon modeling the flooded area. The rest of the data about demands has been extracted from the Experience Feedback (EF) of the rescue teams. However, they are not complete so data following tendencies observed on EF needs to be generated. Data is generated as follows:

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

| Priority | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Deadline (hours) | 6 | 12 | 24 | 24 |

**Table 1. Variables**

- **Action Time**: Uniform distribution on the interval $[300, 2100]$ in seconds. This interval was given by the rescue teams from SDIS 31.
- **Category**: Categories of the demands are distributed proportionally to data extracted from EF.

The demand sizes are depending on the QDI. All the data are generated based on a total number of victims over all the nodes of the category. This number is extracted from EF and scaled to the size of the problem. To generate the quantity distribution series, several laws have been used:

- For low QDI, a truncated normal law is used. The mean equals $a \cdot \frac{nbvictims}{2}$ and the standard deviation equals $b \cdot \frac{nbvictims}{4}$. The best value for $a$ and $b$ to cover the range of quantity distribution are $a = 1$ and $b = 1.6$
- For high QDI, the distribution is based on a truncated normal law with characteristics meant to fit demand sizes extracted from Luchon crisis EF. In this case, the distribution is truncated so that for all x-value: $a \leq x \leq b$, the mean of the distribution is equal to $\mu$, and the standard deviation is equal to $\sigma$. Details are given in table 2.
- For medium QDI, a truncated normal law of mean $a \cdot \frac{nbvictims}{nbnodes}$ and standard deviation of $b \cdot nbvictims$ is used. Values of $a$ and $b$ have been set to $a = 0.2$ and $b = 1$ by experimentation in order to complete the specter of QDI values.

| 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 |

**Table 2. Distribution laws for high QDI values series generation**

Figure 5 is an example of a data set for quantity distributions generated by a truncated normal law. In this figure, the quantity is displayed on the x-axis and the rate of total nodes on the y-axis. It shows data series for a highly distributed scenario.

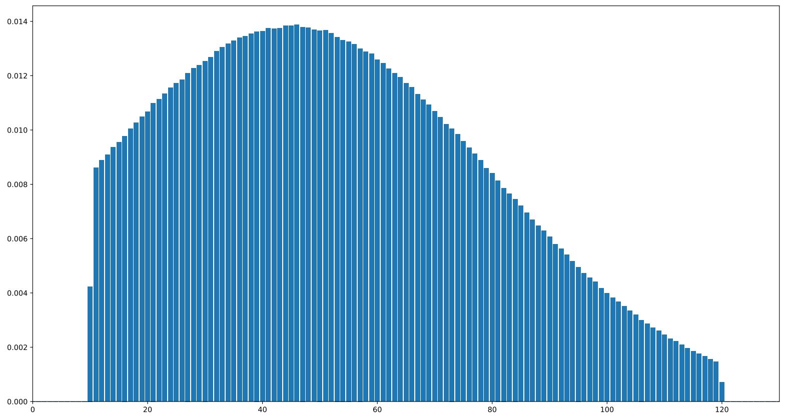


**Figure 5. Example of a truncated Normal to generate Quantity distribution**

Using these laws 10000 data series are drawn per law, sorted by QDI and stored in a data set. This process allows covering the full specter of QDI values. When a quantity distribution series is needed for the experiments, it is pulled from this set.

The same process is executed for the generation of time series associated with a given dynamism. For the dynamism different generation laws are used:

- For low dynamism time series, a Non-Homogeneous Poisson process is used with a rate function $\lambda(t) = a \cdot \sin(t \cdot f \cdot 2\pi - p \cdot \pi) + h$. The value of the parameters have been tuned in order to generate time series in the desired dynamism range, the following parameters values have been set. The amplitude $a = 1.05$, the frequency $f = 2.5 \cdot 10^{-4}$, the phase $p$ is generated with a uniform law between 0 and 1 and $h$ between $-0.99$ and 1.5.
- For high dynamism time series, a uniform law is used with a lower bound $lB$ and a higher bound $hB$ computed at each generation according to the time horizon $T$ and the number of nodes $V$. Then $lB = \frac{T}{V} - dvt$ and $hB = \frac{T}{V} + dvt$ with $dvt$ generated randomly by a normal law of mean $0.9 \cdot \frac{T}{V}$ and standard deviation $1.2 \cdot \frac{T}{V}$.

The fig. 6 shows time series dispatch on the dynamism range by number of time series on ordinates for each dynamism value in abscissa for 10000 time series generated per law. The coverage of the dynamism specter of each law is competitive on the transition values but it allows to ensure full coverage of all values for dynamism desired.

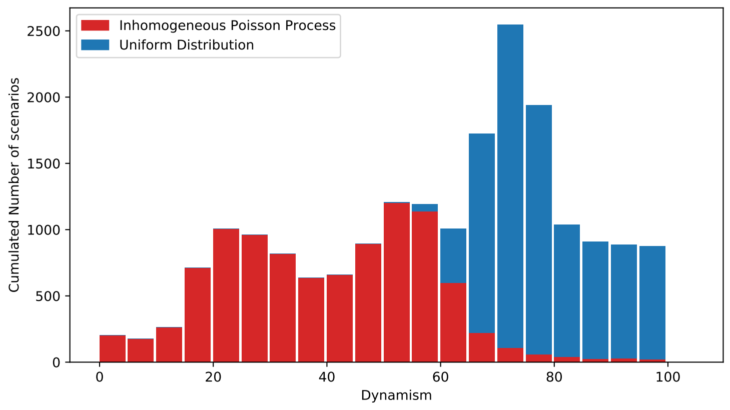


**Figure 6. Dispatching of time series by dynamism according to the generation process**

For the rest of the paper, given a value of QDI or dynamism, the data series is selected for an interval of size 5 around this value. The last data series that need to be generated concern the other type of dynamic events: the delays, more precisely their values and release dates of delays. Values have been obtained after discussions with the rescue teams. For the travel time delays, the value is uniformly distributed between 60 and 900 seconds. The action time delays values are picked between 10 and 120 seconds and are multiplied by the number of victims rescued during the mission delay. According to the SDIS 31, 15% of the missions are delayed either on travel or service.
For the release dates, precise dates cannot be simply generated in advance since we do not know if a mission to the node will be in progress at this date. Furthermore, in the perspective of comparing algorithms on the same basis, the release time generation process needs to be 'fair'. For that purpose, a table of delays is generated for every node. This table is constituted of one value for each period of 10 minutes, for the entire length of the time horizon. And since the rate for delays is 15%, each value of as a 15% of being non-null and pulled from the ranges mentioned above. Otherwise, the value in the table is null signifying there are no delays for these 10 minutes.
When a mission is executed at a node, the value of the delay is read in the table at the time period column corresponding to the planned date of the arrival at the node.

It is necessary to have a tool to run the Dynamic scenarios for the evaluation process. A homemade simulator is presented below.

## Simulator

In this article, simulation is used to run crisis scenarios generated in advance. Thanks to the simulator, we are able to run scenarios in simulated time to speed up the evaluation process. The simulator plays the different events that would happen in real life, and calls the heuristics to find routes, then it communicates routes from the DC to the vehicles and events from vehicles to the DC. In a real-life crisis, the solution would be used directly by the decision center's operators to help build rescue vehicle routes; the simulated part would be removed, the heuristics would be directly used by operators with some inputs (demands, priorities, vehicles...).

The simulator was developed to answer several requirements for the dynamic experiments:

- Simulate the communications process between the DC and the vehicles during the crisis.
- Run a crisis scenario in simulated time to avoid time-consuming experiments in real-time. Thanks to simulated time, one may run a crisis of several hours in a few minutes.
- Record and log different metrics during the crisis to evaluate the solutions' qualities afterward.

The simulator is built as a multi-process application where each vehicle is represented with a different process as well as the DC. An ActiveMQ bus is integrated to enable the communication between the processes through different message formats as shown in figure fig. 7.

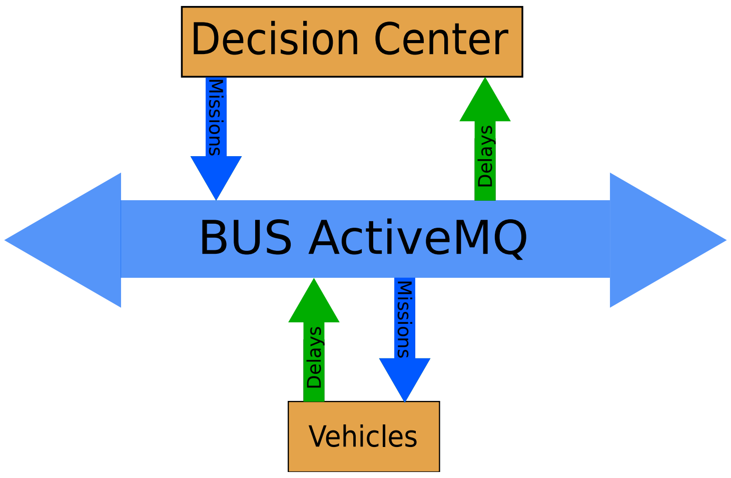


**Figure 7. Processes communication with ActiveMQ Bus**

To ensure the synchronization of these processes and their communications, a metronome is used. This component of the simulator also enables to play the scenarios in simulated time. Every process gives its date for its next event (new request for the DC and arrival to a node or end of vehicle service) to the metronome and the metronome wakes the processes in chronological order. Once a process is awakened, it treats the dynamic event, gives its next date to the metronome, and goes back to sleep. The treatment of the events depends on the type of process and event concerned:

- Vehicles: Vehicles can treat arrival to nodes or end of service to nodes. In both cases, the value of the delay is computed according to the arrival date at the node and the delays table. The delay is sent to the DC. If the event is the end of a service, the vehicle waits for the response from the DC with its next mission to execute. Else the vehicle goes back to sleep until the date of the end of the service at the current node.
- DC: This process implements all the decisions to be taken. It treats received delays and new requests. The DC synchronizes on the next new request in the data series. The synchronization can be interrupted by the DC's listener when it receives delay. In this case, the DC wakes up, treats the delay, and synchronizes on the next release date. In all cases, the treatment is either the computation of a new plan or the bufferrisation of the event depending on the strategy.

When there are no more new requests and when all the missions have been served by the vehicles, the simulation ends.

## SOLUTIONS

When a dynamic scenario is run, algorithms have to be used at different dates to update the plans as explained above. Several algorithms are presented in this section.

### Re-optimization

According to the dynamic of the crisis, the time to compute a new plan including dynamic events is too short to consider exact methods. Since static heuristics have been evaluated with goods performances on the static problem, the re-optimization using these heuristics is developed. In this section the static heuristics Best Flow-time Insertion (BFI) and Best Flow-time Insertion with Order Questionning (BFIOQ) are presented. These static algorithms are adapted to a re-optimization scheme. The mechanism implemented for this adaptation is presented in this section.

BFI is an insertion heuristic similar to the Best Fit allocation scheme from the bin packing problem literature. In this algorithm, the demands are sorted and inserted at the best position in the tours given defined c riteria. The demands are sorted by priority and then by decreasing size. This way the demand served first is the most likely to put higher constraints and impact on the objective score. This role is operated through a *sortDemands* function in the algorithm. The insertion criterion relies on the objective function (1). Flow-time Insertion Score (FIS) represents the impact an insertion has on the objective function. It is computed for the insertion in the plan of vehicle $k$ to serve node $j$ after serving node $i$ as so:

$$FIS(i,j,k) = \frac{p_j \times (tt_{i,j,c} + a_i + h^z_{i,k})}{q^z_{j,k}}, \quad \forall z \in \mathcal{Z}, c \in C \tag{10}$$

If the insertion leads to a deadline violation, the function returns $+\infty$. The simplified algorithm can be presented as:

```
$queue\_of\_demands \leftarrow sortDemands(\mathcal{V}^\star)$
for $cat \in categories$ do
    for $dem \in queue\_of\_demands$ do
        for $vehicle \in \mathcal{M}$ do
            for $pos \in \{1, \dots, planSize(vehicle\}$ do
                $score \leftarrow FIS(pos, dem, vehicle)$
                if $score < bestScore$ then
                    $bestScore \leftarrow score$
                    $bestVehicle \leftarrow vehicle$
                    $bestPosition \leftarrow pos$
                end if
                if $\exists bestVehicle \& bestPosition$ then
                    $insertDemand(bestScore, bestVehicle, bestPosition)$
                else
                    $DeadlineViolation$
                end if
            end for
        end for
    end for
end for
```

With functions:

- *planSize*: This function computes the available and valid position where the current demand might be inserted. It might be at the beginning of the current tour of the vehicle, at its end, or between two already inserted demands.
- *insertDemand*: This procedure inserts the demand in the plan once the best vehicle and position to insert have been defined. Once it has done the insertion routine, it also updates the current tour if needed.
- *DeadlineViolation*: This procedure is used when during the execution of BFI or BFIOQ no vehicle is a valid candidate for insertion without violating the deadline for the current demand. More details are given below.

In the case *DeadlineViolation* is called, all computation is dropped since the deadline violation is related to previous insertions. In this case, Earliest Deadline First (EDF) algorithm is used to find a solution that could not violate the deadlines. EDF is a scheduling algorithm introduced as Deadline Driven algorithm by Liu and Layland 1973. This algorithm as its name indicates, insert demands into the plan by order of deadline. Since several vehicles are available, the demand with the earliest deadline is inserted at the end of the route of the first available vehicle, in consideration of the state of the plan at the moment of the insertion. If a suitable solution is found, the heuristic returns the solution from EDF. The deadline is considered violated otherwise and an error is logged. In fact, if both BFI and EDF cannot manage to compute a valid solution, it is considered that the problem might be infeasible and human decision is required.

BFIOQ is only different from BFI from a routine launched after each insertion through a call to $insertDemand$. If the current turn of the vehicle already contains at least one other intervention, the order in which the demands are served is questioned using a brute force algorithm. Then the computation of the solution continues as described in BFI.

Since this algorithm is used in a re-optimization scheme, adaptations need to be made from the static version presented above. The algorithm has been adapted with a new parameter representing the current state of the crisis: current position of each vehicle, current load state (number of victims in the vehicles), and mission in progress at the time of re-optimization. This state parameter is updated each time a mission is attributed to a vehicle since it is considered in this work that once a vehicle starts the travel, the demand being currently served is not modified and the vehicle is not re-routed. The heuristics have been adjusted to begin planning starting from the current situation and avoiding questioning the currently served demand.

### Insertion Heuristic

Insertion heuristics show good results in the literature when computation time is an important issue of the problem. This category of heuristics only inserts the new demand in the existing plan instead of recomputing the entire plan with the new demand. For example Angelelli et al. 2009 and Albareda-Sambola et al. 2014 use such methods. A strategy from Potvin et al. 2006 has been studied. They present an algorithm that inserts demands at the end of the plan when they are released. The solution has been adapted and improved to insert demands at a different position in the plan, not only at the end.

This insertion heuristic Flow-time Insertion Phase-out (FIP) uses an adaptation of the FIS presented in the section above to compute the impact on the objective score to insert the new demand at the position of a planned mission.

For the insertion of demand at node **i** for vehicle **k** on tour **z** at the position of mission **m** of node **m.node**, the Flow-time Insertion Phase-out Score (FIPS) is expressed as follows:

$$FIPS(i,m,k,z) = p_i \cdot h^z_{i,k} - p_j \cdot h^z_{m.node,k} + \sum_{n \in nextNodes} p_n \times (h^z_{i,k} - h^z_{m.node,k}), \tag{11}$$

where *nextNodes* is the list of all the missions in the plan after the phased-out mission.

These scores take into account the cost of removing the mission from the plan so the benefits from inserting the new demand have to be worth phasing out the already planned mission. The objective is then to lower the score. The mission with the lowest negative FIPS is then selected and the replacement is applied. The phased-out mission is then placed in a queue of demands by priority order. The process is repeated as long as the queue is not empty. If none of the planned missions is a fit for insertion, meaning no mission gives a negative FIPS, then a score is computed for insertion at the end of the plan for each vehicle. The vehicle with the lowest score is selected. Insertion in this vehicle is processed to fill the last effective tour and then another best vehicle for insertion at the end is selected. This routine is repeated until the demand is served entirely. The algorithm might be simplified as:

```
$queue\_of\_demands \leftarrow newDemands$
while $queue\_of\_demands \in categories$ do
    $i \leftarrow queue\_of\_demands[0]$
    $list\_of\_missions \leftarrow computeListOfMissions$
    $bestScore \leftarrow 0$
    while $q_i \neq 0$ do
        for $m \in list\_of\_missions$ do
            if $FIPS(i, m, m.vehicle, m.turn)$ then
                $bestScore \leftarrow FIPS(i, m, m.vehicle, m.turn)$
                $bestMission \leftarrow m$
            end if
        end for
        if $bestScore \neq 0$ then
            $insertMission(bestMission) \leftarrow computeListOfMissions$
        else
            $insertAtTheEnd(m)$
        end if
    end while
end while
```

With the functions:

- *computeListOfMissions*: Computes the missions from the plan and returns a list of the missions, candidates to be phased-out.
- *insertMission*: Replace a mission in the plan and add the phased-out mission in the queue of demands. The quantity left to serve the demand inserted is updated.
- *insertAtTheEnd*: Selects the first available vehicle according to the plan already set and inserts all the possible quantity to complete the last turn of this vehicle

In case a deadline violation is detected on an insertion (no insertion candidate offers negative FIPS without deadline violation), the score evaluation/insertion routine is re-launched but this time not looking for a benefit on the FIPS and no deadline violations. If after this routine, no valid candidate is found, the algorithm returns a deadline violation.

## EXPERIMENTAL RESULTS

The initial value for RCP is determined according to the size of the problem. Experiments have been led with 360 scenarios of various characteristics on the size of the case of study (60 points of demand). The mean of the computation time for the BFIOQ algorithm, that is theoretically the more complex algorithm, overall computation (more than 50 per scenarios) over all the scenarios have been born up to 4 seconds. To size this value to the problem's scale, the quadratic complexity of the BFIOQ algorithm is used. For example for a problem of size 180 nodes, the initial RCP is calculated as follows:

$$RCP(180) = RCP(60) \times \left(\frac{ProblemSize}{60}\right)^2 = 4 \times \left(\frac{180}{60}\right)^2 = 36\ seconds \tag{12}$$

The value for the initial RCP could be selected with any value consequent with characteristic computation time during the crisis but the advantage of selecting it from a set of experiments is to reduce the number of overtimes by being more accurate at the beginning of the crisis.
The experimental set is based on Luchon's flooding data. Therefore the scale is set accordingly with 15 initial nodes and 45 dynamic requests for a total of 6 nodes to serve. The dynamic requests are released over a time horizon of 4 hours. The experimental set is built on 6 dynamism values from 0 to 100 with a step of 20, 6 QDI values from the same range, and for each pair of (dynamism, QDI), 10 scenarios are generated following the above process. This makes a total of 360 scenarios in the set.
The set is used to evaluate the three algorithms presented in this article each associated with the three strategies for a total of 9 combinations. The simulator has been dockerized and the simulations are conducted on nodes of the platform Grid'5000, a french large-scale academic platform for experiment-driven research in all areas of computer science. Experiments with 6 containers in parallel have been driven on nodes with 2 CPUs Intel Xeon E5-2630 v3 of 8 cores and 128 GiB of memory.

Figure 8 displays the evolution of the objective score according to the dynamism for all 9 combinations of algorithms and strategies. As specified in the legend, the colors represent the algorithms and the shapes of the markers the strategy used. The objective score is one of the main criteria of comparison taken into account for this evaluation section. It is the value of the objective function considering all the demands served during the duration of the simulation. For this figure, the mean of the objective score is computed on all the scenarios of the given dynamism, whatever the QDI. Consequently, every point of the figure represents 60 scenarios. The results of this figure as well as all the graphics in the evaluation section have been normalized by the best algorithm/strategy scenario by scenario when plotting the flow-time score.

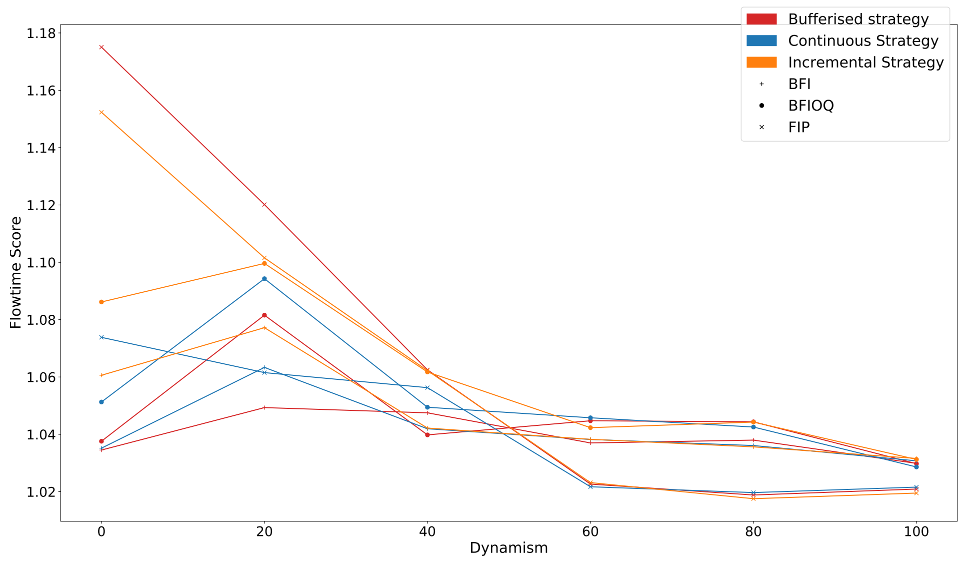


**Figure 8. Objective score normalised by the best in function of Dynamism**

The results do not show a clear superiority of any of the algorithms or strategies from this perspective and the differences of objective scores are very low for one algorithm and strategy from a nother. However a tendency is observable where BFI and BFIOQ show similar trend unlike FIP. In fact, FIP seems to show better performances than the other algorithms for high dynamism whereas for low dynamism values BFI and BFIOQ show better performances. Figure 9 confirms that by displaying the same results and merging all the strategies for each algorithm.

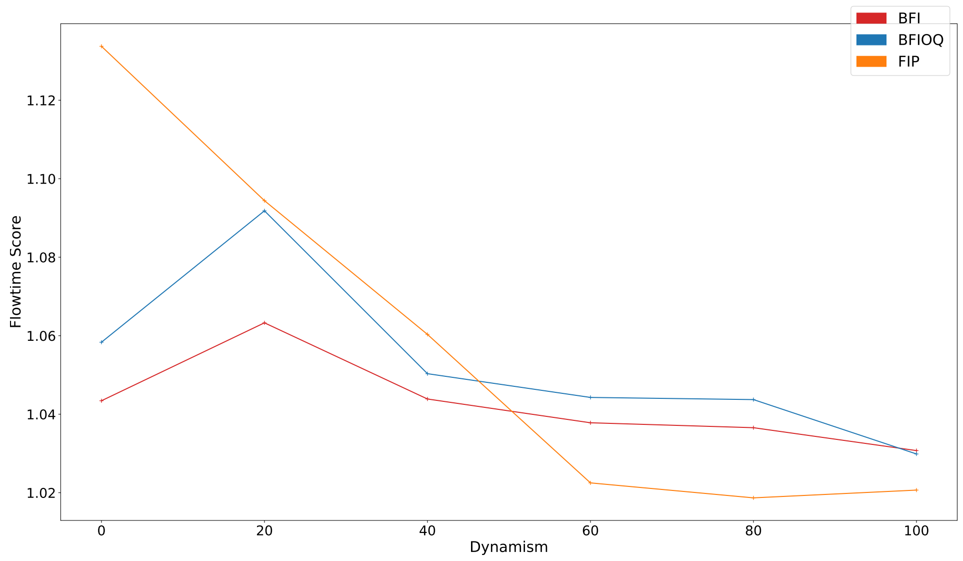


**Figure 9. Objective score normalised by the best algorithm in function of Dynamism Merging strategies to display algorithms**

On the opposite, when trying to display these results by merging all algorithms for each strategy like in fig. 10, one can observe a small advantage for the Continuous Strategy over the Buffered Strategy which itself shows better results than the Preemptive strategy.

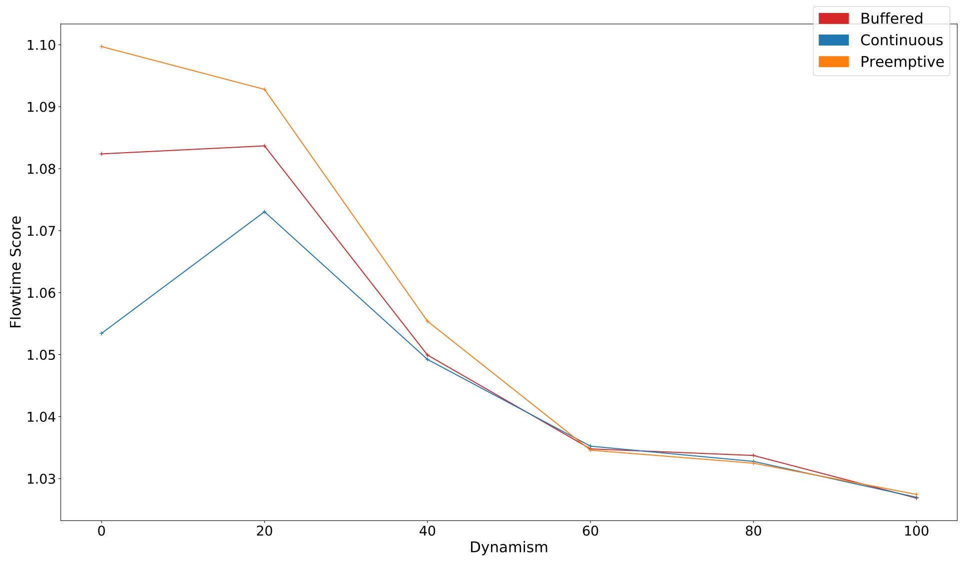
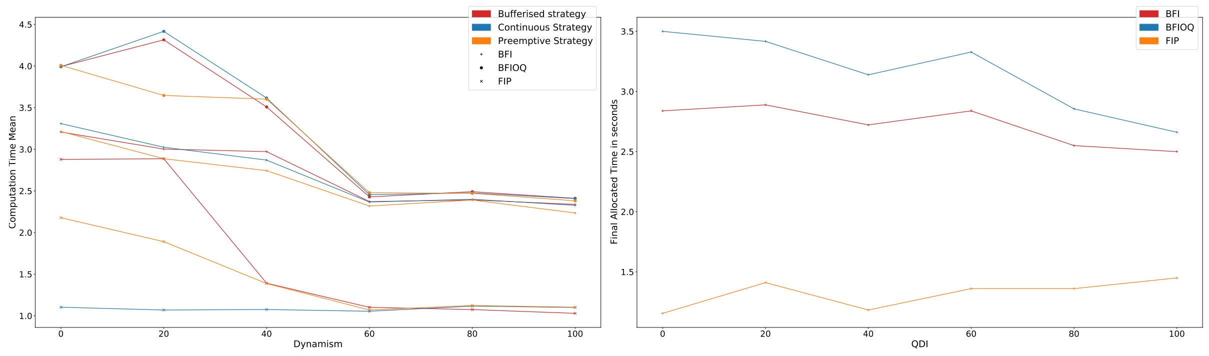


**Figure 10. Objective score normalised by the best strategy in function of Dynamism Merging algorithms to display strategies**

These results show that BFI and BFIOQ have better performances at low dynamism compared to FIP. At low dynamism, the new requests are released by batch: several demands in a short amount of time. BFI and BFIOQ re-compute all the solutions when re-optimizing, when the dynamism is low, there are more demands to question and therefore more possibilities of gain for these algorithms compared to FIP that computes a gain one by one. On the opposite when the dynamism is high, the release dates are more equally distributed over the time horizon, which leads to fewer opportunities for gain for BFI and BFIOQ. On the contrary FIP is not concerned by this drawback at high dynamism and shows better results in term of computation time as displayed in fig. 11a, which allows it to close the gap and even be better may be due to other factors.

The fig. 11b displays the mean RCP at the end of the simulation which corroborates the results of the computation time performances displayed in fig. 11a and might have less negligible impact at higher scale. In fact, since we time the RCP by 2 at every overtime, the final RCP is the initial RCP times the number of overtimes power 2.



**(a) Computation time normalised by the best** **(b) RCP at the end of the simulation normalized by the best**

**Figure 11. Graphics of the evolution of errors over 30 scenarios in function of problem size**

In fig. 10 mentioned above, results also show better performances at low dynamism for the Continuous strategy which is coherent since it is the strategy that updates the solution the most often. Then comes Buffered strategy that sometimes waits before re-optimizing and finally Preemptive strategy that might interrupt re-optimizations and lead to serve demands later. These trends are obviously put forward at low dynamism since release dates are more likely to be close increasing the impact of computation time in regards to time between re-optimizations. At high dynamism, the release dates are more distributed and as observed there are no clear results for high dynamism values. These results confirm the results from VanLon et al. 2016 that stated that dynamism influence the quality of solutions. BFI and BFIOQ keep satisfying performances at high dynamism whereas FIP is very bad at low dynamism. This motivates the choice to use the re-optimization heuristics in all scenarios of this scale.

The other metric (QDI) has also been evaluated but no tendency has been observed on its influence on the solution's qualities.

For the evaluation process, if an algorithm does not achieve to find a solution without violating deadlines during the simulation, the scenario is counted as an error.

The scenario is then removed from the pool of scenarios for the evaluation. However, this case has not been encountered during this experimentation. The scale of the experiments was not big enough or the resources too comfortable to challenge the algorithm sufficiently to conduct to deadline violations er rors. To evaluate the resilience of the algorithms to deadline constraints, another experimental protocol has been built. The objective is to set values of dynamism and QDI but change the size of the problem without increasing the available resources. The values for dynamism have been set to 80 and the QDI to 60, values for which the former results showed the clearer tendency. fig. 12 displays the evolution of the number of errors (over 30 scenarios) by a lgorithms. The figure shows that FIP produces more than 2 times more errors when the size reaches 100 nodes and more. This makes sense since FIP tests fewer possibilities for insertions of new demands and questions less the current plan so in the long term this leads more likely to deadline violation situations. The strategy does not have clear differences toward resilience to deadline constraints.

In conclusion, FIP developed as an improved strategy from the literature have shown better performances in terms of computation time compared to the algorithms BFI and BFIOQ. It also shows slightly better results for scenarios with high dynamism and low quantity distribution. However, the differences are very little in the situation where it has the advantages. In the other situations, BFI and BFIOQ show higher performances even if they are slower to compute an updated solution. Furthermore, these algorithms are more resilient to constraints which means they are more likely to produce a valid solution even in difficult si tuations. However, it would be interesting to study an higher scale of problems with also scaling up the resources. In this context, the algorithm might have enough resources to avoid deadline violations contrary to the second experiment, but the computation time would probably increase and the inter-arrival time reduced which could give an advantage to the algorithm like FIP that have lower computation time.

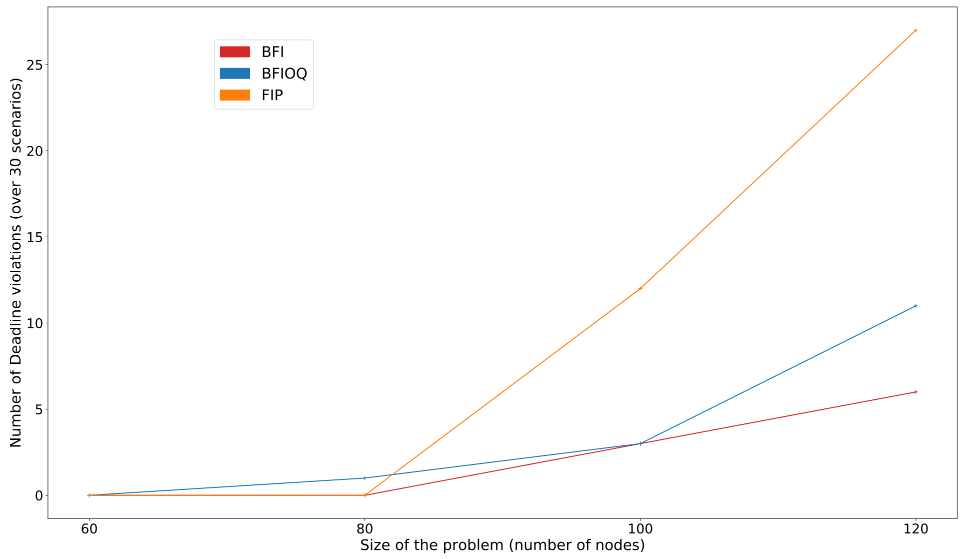


**Figure 12. Comparison of algorithms errors merging strategies to display algorithms**

## CONCLUSION

This article presents a model for the short-term problem of victim relief operations faced by rescue teams during flash flood events. The dynamic aspect of the problem is developed and 3 algorithms have been introduced. On the one hand, BFI and BFIOQ, which have been developed originally for the static version of this problem, are adapted to solve dynamic problems. On the other hand, FIP is based on insertion heuristics from the literature and is adapted to the constraints of this problem. These 3 heuristics are adapted for re-optimization approaches that are best fitted to solve problems with these characteristics. In this context, three strategies for re-optimization have been presented: Buffered, Continuous, and Preemptive.

To evaluate these algorithms and strategies in a dynamic context, a graph generator has been developed. It generates dynamic scenarios with a controlled dynamism and QDI, a metric that evaluates victims' proportion distribution among demands.

A multi-process simulator is also presented to play crisis scenarios in simulated time. Simulations have then been run to evaluate the performances of the algorithms and strategies facing various types of dynamic scenarios. Results highlight that BFI and BFIOQ offer better performances and resilience to deadline constraints. However, the computation time is higher than heuristic like FIP and could lead to considering new demands later but the impact was not significantly observed at this scale of experiments.

The model might be generalized to other emergency operations such as the delivery of foods and supplies to impacted areas or the interventions of specialized teams on electrical infrastructures for example. Future experiments are to be conducted to evaluate the results at a higher scale, which could lead to different observations with a more degraded situation and more demand to serve in the same time horizon

## ACKNOWLEDGEMENT

The work presented in this article is funded by the French Research Agency under the project e-Flooding (ANR-17-CE39-0011). This work was partly supported by the French Research Agency under the project Datazero 2 (ANR-19-CE25-0016). Special thanks are addressed to Amal Sayah that worked on the Simulator developed for Datazero 2 project and gave his time to help adapt this simulator for this work.
Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities as well as other organizations (see https://www.grid5000.fr).

## REFERENCES

Albareda-Sambola, M., Fernández, E., and Laporte, G. (2014). "The dynamic multiperiod vehicle routing problem with probabilistic information". In: Computers & Operations Research 48, pp. 31–39.

Angelelli, E., Bianchessi, N., Mansini, R., and Speranza, M. (2009). "Short Term Strategies for a Dynamic Multi-Period Routing Problem". In: Transportation Research Part C: Emerging Technologies 17.2, pp. 106–119.

Archetti, C., Feillet, D., Mor, A., and Speranza, M. (2020). "Dynamic traveling salesman problem with stochastic release dates". In: European Journal of Operational Research 280.3, pp. 832–844.

Bent, R. and Van Hentenryck, P. (Aug. 2003). "Dynamic Vehicle Routing with Stochastic Requests". In.

Bent, R. W. and Van Hentenryck, P. (2004). "Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers". In: Operations Research 52.6, pp. 977–987.

Booij, M. (2005). "Impact of climate change on river flooding assessed with different spatial model resolutions". In: Journal of Hydrology 303.1, pp. 176–198.

Gendreau, M., Guertin, F., Potvin, J.-Y., and Séguin, R. (2006). "Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries". In: Transportation Research Part C: Emerging Technologies 14.3, pp. 157–174.

Gendreau, M., Laporte, G., and Semet, F. (2001). "A dynamic model and parallel tabu search heuristic for real-time ambulance relocation". In: Parallel Computing 27.12, pp. 1641–1653.

Golden, B. and Assad, A. (1986). "Vehicle Routing with Time Window Constraints". In: American Journal of Mathematical and Manag 6.3-4, pp. 251–260.

Ilgaz, S., Fernando, O., and MagedDessouky (2008). "A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty". In: IIE Transactions 40.5, pp. 509–523.

Laporte, G. (1992). "The vehicle routing problem: An overview of exact and approximate algorithms". In: European journal of operational research 59.3, pp. 345–358.

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

Liu, C. L. and Layland, J. W. (1973). "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment". In: Journal of the ACM 20.1, pp. 46–61.

Mitrović-Minić, S., Krishnamurti, R., and Laporte, G. (2004). "Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows". In: Transportation Research Part B: Methodological 38.8, pp. 669–685.

Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013). "A review of dynamic vehicle routing problems". In: European Journal of Operational Research 225.1, pp. 1–11.

Potvin, J.-Y., Xu, Y., and Benyahia, I. (2006). "Vehicle routing and scheduling with dynamic travel times". In: Computers & Operations Research. Part Special Issue: Optimization Days 2003 33.4, pp. 1129–1137.

Ritzinger, U., Puchinger, J., and Hartl, R. F. (2016). "A survey on dynamic and stochastic vehicle routing problems". In: International Journal of Production Research 54.1.

Stolf, P., Pierson, J.-M., Sayah, A., Da Costa, G., and Renaud-Goud, P. (Jan. 2019). "e-Flooding: Crisis Management through Two Temporal Loops". In.

Tramblay, Y. and Somot, S. (2018). "Future evolution of extreme precipitation in the Mediterranean". In: Climatic Change 151.1, pp. 289–302.

Van Hentenryck, P., Bent, R., and Upfal, E. (June 2010). "Online stochastic optimization under time constraints". In: Annals of Operations Research 177.1, pp. 151–183.

VanLon, R. R. S., Ferrante, E., Turgut, A. E., Wenseleers, T., Vanden Berghe, G., and Holvoet, T. (Sept. 2016). "Measures of dynamism and urgency in logistics". In: European Journal of Operational Research 253.3, pp. 614–624.

Vinet, F., Lumbroso, D., Defossez, S., and Boissier, L. (2012). "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". In: Natural Hazards: Journal of the International Society for the Prevention and Mitigation of Natural Hazards 61.3, pp. 1179–1201.

Vu, D. M., Hewitt, M., Boland, N., and Savelsbergh, M. (2020). "Dynamic Discretization Discovery for Solving the Time-Dependent Traveling Salesman Problem with Time Windows". In: Transportation Science 54.3, pp. 703–720.

Wen, M., Cordeau, J.-F., Laporte, G., and Larsen, J. (2010). "The dynamic multi-period vehicle routing problem". In: Computers & Operations Research 37.9, pp. 1615–1623.

Yang, J., Jaillet, P., and Mahmassani, H. (2004). "Real-Time Multivehicle Truckload Pickup and Delivery Problems". In: Transportation Science 38.2, pp. 135–148.

Yang, W.-H., Mathur, K., and Ballou, R. H. (2000). "Stochastic Vehicle Routing Problem with Restocking". In: Transportation Science 34.1, pp. 99–112.