

The vehicle routing problem for flash floods relief operations

Florent Dubois

▶ **To cite this version:**

Florent Dubois. The vehicle routing problem for flash floods relief operations. Modeling and Simulation. Université Paul Sabatier - Toulouse III, 2022. English. NNT : 2022TOU30074 . tel-03726667

HAL Id: tel-03726667
https://theses.hal.science/tel-03726667

Submitted on 18 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *05/04/2022* **par :**

Florent DUBOIS

The Vehicle Routing Problem for Flash Floods Relief Operations

---

JURY

Andréa CYNTHIA SANTOS	Professeur d'Université	Rapporteuse
Jean-Marc NICOD	Professeur d'Université	Rapporteur
Jean-Marc PIERSON	Professeur d'Université	Examinateur
Caroline Rizza	Maître de Conférence	Examinatrice
Frédérick Benaben	Professeur d'Université	Invité
Marie-José HUGUET	Professeur d'Université	Invitée
Paul RENAUD-GOUD	Maître de Conférence	Co-encadrant de Thèse
Patricia Stolf	Maître de Conférence	Directrice de Thèse

École doctorale et spécialité :
MITT : Domaine Mathématiques : Mathématiques appliquées

Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)

Directeur(s) de Thèse :
Patricia STOLF et *Paul RENAUD-GOUD*

Rapporteurs :
Andréa CYNTHIA SANTOS et *Jean-Marc NICOD*

Résumé

Chaque année, les inondations ont de graves conséquences en termes de pertes humaines ainsi que de dégâts sur les infrastructures. Le projet ANR i-nondations s'intéresse à ce problème et étudie des moyens d'amélioration des phases de réponse et de récupération de la gestion de crise. Ce projet se concentre plus précisément sur les crues éclairs, un type d'inondation caractérisé par une augmentation très rapide du niveau de l'eau. Cela ne laisse que très peu de délai aux équipes de secours pour des mesures préventives. Le travail présenté dans cette thèse se concentre sur la phase de réponse aux crues éclairs.

En collaboration avec les équipes de secours du SDIS 31 (*Service Départemental d'Intervention et de Secours*), les opérations de secours aux victimes, afin de les conduire en lieu sûr sont étudiées.

Le problème étudié est l'optimisation des trajets des véhicules de secours entre des lieux où des victimes sont en péril. Une fois une victime secourue, le véhicule doit l'amener en sûreté. Ce type de problème appartient à la catégorie des Problèmes de Tournées de Véhicules. Un enjeu majeur du problème est la limite de capacité des véhicules. Cela doit être pris en compte lors de la planification des trajets des véhicules. Certaines demandes peuvent nécessiter d'être secourus par plusieurs véhicules car il y a trop de victimes à prendre en charge pour qu'un véhicule puisse le faire seul. De plus, les interventions sont priorisées. En fonction des victimes à secourir, l'intervention peut avoir un caractère plus ou moins urgent. Cela dépend de l'âge, de l'état de santé ou encore du lieu où se trouve une victime ainsi que de nombreux autres facteurs. En effet une personne âgée ne peut pas attendre d'être secourue aussi longtemps que le peut un jeune adulte par exemple. À ce critère de priorité est également associée une deadline : la date la plus tardive à laquelle une intervention doit avoir lieu. De plus, les équipes de secours du SDIS 31 utilisent différentes catégories de véhicules pour leurs interventions. Ces véhicules ont des capacités différentes et peuvent gérer différents types d'interventions. Ces contraintes ajoutent de la complexité par rapport au problème général de Tournées de Véhicules. Cependant, les équipes de secours opèrent souvent dans un contexte dégradé, ce qui signifie qu'elles ne possèdent pas suffisamment de ressources pour secourir toutes les victimes en un seul trajet. Leurs trajets sont donc planifiés sur plusieurs tours. Un tour étant comptabilisé à chaque fois qu'un véhicule doit rentrer au centre de secours pour mettre les victimes en sécurité. Ainsi, il recommence un tour avec sa capacité totale. En connaissance de cause, diviser les interventions entre plusieurs véhicules est autorisé dans le problème étudié dans cette thèse. Cela veut dire que les trajets des véhicules peuvent être planifiés pour secourir des victimes à un lieu donné avec plusieurs interventions, ce qui donne plus de flexibilité aux équipes de secours par rapport au problème général de Tournées de Véhicules.

Premièrement, le travail de cette thèse est de trouver des solutions au Problème de Tournées de Véhicules rencontré par les équipes de secours et le mettre en relation avec des travaux similaires de la littérature de *gestion de crise* et de la littérature des Problèmes de Tournées de Véhicules. La première contribu-

tion est d'étudier la version statique du problème. Des approches diverses sont développées et comparées. Une approche de résolution par méthode exacte est testée et ses performances sont mesurées en fonction de l'évolution de la taille du problème (nombre de lieux d'intervention). Des algorithmes heuristiques sont développés afin de pouvoir traiter les problèmes de grande échelle. Enfin, le problème de répartition des ressources que rencontrent les équipes de secours est étudié. Quand une inondation impacte plusieurs zones géographiques à la fois, les équipes de secours peuvent avoir besoin de répartir leurs véhicules sur les différents secteurs impactés pour assurer les opérations en tous lieux. Le problème d'optimisation en question est étudié. Des solutions diverses sont ainsi développées et comparées dans cette thèse.

Pendant une inondation, les équipes de secours n'ont pas toutes les informations sur les lieux d'interventions dès le début de la crise. Des évènements dynamiques de natures variées se révèlent au cours de la crise pendant que les équipes de secours sont déjà en route. Les lieux où les victimes doivent être secourues, ainsi que le nombre de victimes à prendre en compte, peuvent être révélés à la volée grâce au repérage de la zone sinistrée par exemple. De plus, les équipes de secours opèrent dans un contexte dégradé, ce qui peut induire des détours par rapport aux trajets prévus et des délais par rapport au planning. Par conséquent une autre classe de problème est étudiée quand des informations sont dynamiques. Ce problème est appelé Problème de Tournées de Véhicules Dynamique. Des solutions adaptées pour répondre à cette version dynamique du Problème de Tournées de Véhicules sont développées et évaluées dans cette thèse.

Dans le but d'évaluer les solutions à une crise, des outils logiciels ont été développés. Un générateur de graphes est proposé afin de générer des instances de territoires inondés et leurs enjeux. Ce générateur a été développé dans le but de permettre de paramétrer les instances créées. De ce fait, après analyse des données contenues dans les Retours d'Expérience de la crue éclair de Luchon, dans le sud de la France en 2013, qui est le cas d'étude de cette thèse, des instances de test avec des caractéristiques similaires ont été générées afin de réaliser l'évaluation. Un simulateur qui permet d'imiter les interactions entre les différents acteurs de la gestion de crise a également été développé. Il permet de jouer des scénarios de crise dynamiques en temps simulé. Cela aide à accélérer le processus d'évaluation mené sur un grand nombre de scénarios. Ces scénarios ont été générés en utilisant des données du cas d'étude de la même façon que les instances de territoires. Le processus de génération des scénarios dynamiques permet de créer des scénarios aux caractéristiques contrôlées par deux métriques: la métrique du Dynamisme issue de la littérature et une métrique développée dans cette thèse afin de mesurer la répartition du nombre total de victimes entre les lieux d'intervention. Le processus d'évaluation permet d'observer quelle solution développée dans ce travail affiche les meilleures performances en termes de qualité de solutions et de temps de calcul, et dans quelle conditions. Les heuristiques proposées dans les contributions montrent également de meilleures performances que des solutions adaptées de la littérature.

Abstract

Every year, floods have huge consequences both in terms of human casualties and damage to infrastructures. The ANR i-Nondations (e-Flooding) project tackles such crises and studies means to improve both the response and recovery phases of disaster management. This project more specifically focuses on flash floods, a type of flood characterized by a fast rise in the water level. It leaves very limited time to rescue teams for anticipation measures. The work presented in this thesis focuses on the response phase to flash floods. In collaboration with rescue teams from SDIS 31 (*Service Départemental d'Intervention et de Secours*), victim relief operations are studied. More specifically, the interventions to rescue flood victims and evacuate them to safety.

The problem is one of optimizing routes for vehicles between locations where victims need to be rescued. Once a victim is saved, the vehicle needs to drive them back to safety. This problem is called a Vehicle Routing Problem. A major challenge is that rescue vehicles have limited capacities. It has to be considered when planning vehicles' routes. Some locations may require to be visited by several vehicles because too many victims are involved for one single vehicle. Furthermore, interventions can be prioritized. Depending on the victim to rescue, the intervention can be of various degrees of urgency. It depends on its age, form, and location as well as various other factors. An elderly person cannot wait as long as a young adult before its rescue for instance. A deadline is also associated with this priority: the latest date for the intervention to be conducted. In addition, rescue teams from SDIS 31 use different categories of vehicles to conduct interventions after floods. These vehicles have different capacities and can handle different types of interventions. These constraints add complexity to the general Vehicle Routing Problem. However, rescue teams often operate in a degraded state, which means they do not possess enough vehicles to rescue all victims in one tour. Routes for rescue vehicles are then planned on several tours. A tour is counted every time a vehicle has to drive back victims to safety. By doing so, it recovers its full rescuing capacity. In that knowledge, splitting service is allowed in the studied problem in this thesis. It means routes can be planned to rescue victims at a single location with several interventions, which gives more flexibility than the classical Vehicle Routing Problem.

First, the work of this thesis is to develop solutions to tackle the Vehicle Routing Problem encountered by rescue teams and put it into perspective with related work from the disaster management literature and the Vehicle Routing Problem literature. The first contribution is to tackle the static version of the problem. Different approaches are developed and compared. An exact approach is tested and its performance are measured with the evolution of the problem size (number of intervention locations). Heuristic algorithms are also developed to cope with large problem sizes. Finally, the resource distribution problem faced by rescue teams is studied. When a flood impacts several sectors separated by a too large distance, rescue teams may need to dispatch vehicles in sectors

to ensure rescue operations in all locations. The optimization problem of such a resource distribution is presented. Different solutions are introduced in this thesis and compared.

During a crisis such as floods, all information is not known at the beginning of the rescue operations. Dynamic events of various nature are released during the crisis, while rescue vehicles are already on the road. Locations where victims need to be rescued, as well as the number of victims to rescue, may be revealed on-the-fly thanks to rescue teams scouting the impacted area for instance. In addition, rescue teams operate in a degraded context, which can imply detours on the planned route and delays in timing. Therefore, another class of problem is studied when dynamic information is considered, called Dynamic Vehicle Routing Problem. Solutions adapted to answer the dynamic version of the Vehicle Routing Problem are developed and evaluated in this thesis.

To evaluate solutions to a dynamic crisis, tools have been developed. A graph generator is presented in the thesis to generate instances of a flooded territory and its issues. This generator has been developed to be able to parameterize the characteristics of the created instances. Hence, after data analysis of Experience Feedback from the Luchon flash flood in the South of France in 2013, the case study of this thesis, instances with similar characteristics have been generated to conduct an evaluation. A simulator to run interactions between the different rescue team's entities (vehicles, decision center, and call center) has also been developed. It allows us to run dynamic crisis scenarios in simulated time. It helps to speed the evaluation process conducted on a large number of scenarios. These scenarios have been generated using data from the case study for instance creation. The dynamic scenario generation process enables to create scenarios with controlled characteristics based on two metrics. The Dynamism metric from the literature and a metric developed in this work to measure the distribution of the total quantity of victims among the nodes to serve. The evaluation process allows observation of which solutions developed in this thesis gives the best performances in terms of solutions quality and computation time, and in which conditions. The contribution heuristics have also shown better performances than solutions adapted from the literature.

Remerciements

Avant tout, je souhaite remercier mes encadrants de thèse Patricia Stolf et Paul Renaud-Goud pour leur accompagnement et leurs conseils tout au long de mon doctorat tant sur les aspects scientifiques que méthodologiques. Le soutient moral qu'ils m'ont apporté m'a également beaucoup aidé, plus particulièrement dans la période assez particulière de la pandémie de COVID-19.

J'aimerais également remercier Patricia Stolf pour m'avoir accueilli au sein du projet i-Nondations et au sein de l'équipe SEPIA, ainsi que Jean-Marc Pierson qu'elle a remplacé à la direction de l'équipe au cours de ma thèse. J'en profite pour remercier également tous les membres de l'équipe SEPIA que j'ai côtoyés et qui ont rendu l'ambiance de travail agréable et conviviale : Georges, Amal, François, Léo, Morgan, Maël, Julie et Igor.

J'aimerais adresser des remerciements particuliers à Amal Sayah pour sa patience et sa disponibilité pour résoudre les nombreux problèmes techniques que j'ai rencontrés pour adapter le simulateur qu'il a développé pour le projet Datazero à mon problème. J'en profite pour remercier François Thiébolt pour avoir trouvé le temps de m'aider à mettre en place des moyens pour réaliser mes simulations, et Georges Da Costa pour m'avoir guidé vers la plateforme GRID 5000 qui m'a grandement facilité la tâche dans la réalisation de mes expériences.

Je remercie les autres membres du jury et particulièrement les rapporteurs pour leurs retours et leurs commentaires: Andrea Cynthia Santos Duhamel, Jean-Marc Nicod ainsi que Caroline Rizza, Marie-José Huguet et Frédérick Bénaben.

Enfin j'aimerais remercier tous mes proches qui m'ont supporté pendant ces plus de 3 ans et qui m'ont aidé à traverser cette période. Je pense bien évidement à ma copine Claire mais aussi à mes parents Michelle et Thierry, mes frères Fabien et Julien et mes amis Simon, Morgane, Charlotte, Laurent, Victor, William, Thomas, Boris, Brian, Carlota, Eneko, Julie, Adrien et tous les autres, présents au quotidien.

Contents

1 Introduction **9**
- 1.1 Context . . . 9
- 1.2 Contributions . . . 13
  - 1.2.1 Static contributions . . . 13
  - 1.2.2 Dynamic contributions . . . 13
  - 1.2.3 Evaluation tools contributions . . . 14
- 1.3 Plan of the manuscript . . . 15
- 1.4 Ethical discussion . . . 15

2 State of the Art **17**
- 2.1 Disaster management . . . 17
  - 2.1.1 Mitigation . . . 20
  - 2.1.2 Preparedness . . . 20
  - 2.1.3 Response . . . 21
  - 2.1.4 Recovery . . . 23
- 2.2 Vehicle Routing Problem . . . 23
  - 2.2.1 VRP variations . . . 24
  - 2.2.2 Resolution methods for the Vehicle Routing Problem . . . 28
- 2.3 Simulation tools . . . 39
- 2.4 Conclusion . . . 39

3 Experimentation tools and integration **41**
- 3.1 Graph generator . . . 42
  - 3.1.1 Territory model . . . 42
  - 3.1.2 Territory topology . . . 43
  - 3.1.3 Nodes selection . . . 47
- 3.2 Simulator . . . 47
  - 3.2.1 Multi-process . . . 49
  - 3.2.2 Communication . . . 51
  - 3.2.3 Synchronization . . . 53
- 3.3 Conclusion . . . 54

4 Static problem **57**
4.1 Problem description . . . 59
4.1.1 Mixed Integer Linear Programming . . . 62
4.1.2 Complexity . . . 65
4.2 Heuristics . . . 66
4.2.1 Greedy algorithms . . . 67
4.2.2 Solomon's heuristic . . . 69
4.2.3 Best Flow-time Insertion algorithms . . . 70
4.3 Resources distribution problem . . . 75
4.4 Evaluation . . . 76
4.4.1 Data analysis . . . 77
4.4.2 Small instances experiments . . . 79
4.4.3 Luchon-Like experiment . . . 84
4.4.4 Resources distribution experiment . . . 87
4.5 Conclusion . . . 88

5 Dynamic problem **91**
5.1 Problem description . . . 91
5.1.1 Dynamic events . . . 92
5.1.2 Variables . . . 94
5.2 Solutions for the dynamic Vehicle Routing Problem . . . 97
5.2.1 Dynamic mechanisms . . . 98
5.2.2 Routes computation heuristics . . . 99
5.2.3 Continuous local search . . . 106
5.3 Evaluation . . . 108
5.3.1 Metrics for Vehicle Routing Problem . . . 108
5.3.2 Dynamic scenarios . . . 112
5.3.3 Experimental results . . . 115
5.4 Conclusion . . . 127

6 Conclusion **129**
6.1 Static contributions . . . 129
6.2 Dynamic contributions . . . 130
6.3 Evaluation contributions . . . 131
6.4 Perspectives . . . 131
6.4.1 Other approaches from the literature . . . 131
6.4.2 Prioritization . . . 132
6.4.3 Territory resilience evaluation . . . 132
6.4.4 Big data support . . . 133
6.5 Publications . . . 133

Chapter 1

Introduction

Every year, catastrophic events such as floods cause enormous damages to impacted areas and therefore have a huge impact on people and infrastructures. Authors of [99] offer a summary of human casualties and damages to infrastructures caused by flooding events in France from 1983 to 2010. It estimates at least 252 human casualties and 8,3 billion euros worth of damages. The frequency of such events is likely to increase in future decades. Recent research has been conducted on the relations between climate change and the increase in disasters. The article [96] studies the relations between climate change and the frequency of extreme precipitations in the Mediterranean basin. In this article, the strongest studied model shows a 20% increase in such events by the end of the twenty-first century.

1.1 Context

Crisis management around floods is a complex problem. Various actors are involved in different steps of the crisis management process. Each actor uses different databases where information used for their operations is collected. The communication and synergy between these actors are very limited and each one uses its technological tools to improve its operations. Hence there is a real need in the crisis management domain for a unified technology that allows these actors to communicate between the different phases of disaster management and improve the global response to flood crises.

ANR i-Nondations project [94] in which this thesis is conducted, gathers various specialists from the different fields involved in floods. This project tackles the problem of flash floods characterized by a very quick rise in the water level often resulting in a lack of time to put in place anticipation actions.

i-Nondations purpose is to model flash floods in terms of disaster management and impact on the infrastructure, using data collected by technological or human sensors. In addition to human casualties, floods damage infrastructures like the electrical network for instance. It leads to power cuts for the impacted

areas. The project also aims at tackling such issues and includes an energy provider as an industrial partner (Enedis). It has been funded by the French Research Agency (ANR).

Among them, **SDIS 31** (*Service Départemental d'Intervention et de Secours*), the firefighters entity of the Haute-Garonne department (south of France) and rescue teams in charge of the victim relief operations. During a flood crisis, people are trapped by the water and need to be rescued and conducted to safety. This role is operated by firefighters in France. To improve the response to floods, knowing the processes of the rescue teams, and adapting solutions that can be of use to their operations is very important to reduce the impact of a flood in terms of human casualties. Optimization of routes of rescue vehicles between locations where victims need relief, studied in this thesis, can help to improve their operations and it must be in collaboration with rescue teams to develop solutions relevant to their real processes.

The problem studied in this thesis consists in rescuing flash floods victims. At the beginning of rescue operations, the self-evacuation of inhabitants is considered to be terminated. Operations considered by rescue teams in the problem are the evacuations of issues where victims need to be rescued. The studied problem can be modeled as a Vehicle Routing Problem (VRP). In this problem, the objective is to plan optimal routes for a vehicle fleet to visit locations. Rescue teams need to relieve victims at issues locations and drive them to safety. According to the domain where a notion is applied, it is named differently, that is why in the rest of the manuscript, several denominations are used for these locations. Therefore in the VRP literature, they are called *requests* or *demands* but to avoid confusion we will use the term **demand** in the manuscript since the term request will be used to describe another concept. They may also be named *nodes* to fit with the model of the territory as a graph usually used in the literature. When a relief operation is conducted, it is called a *service* in the VRP literature. Hence, "serve a demand" means that victims are rescued at a location.

In the studied problem, requirements must be considered to fit with the operation process of rescue teams:

- **Priorities and Deadlines**: Demands are characterized by a priority. Rescue teams use this metric to help them to order their interventions. In their processes 4 priority levels are defined:

  1. Must be rescued in an emergency
  2. Must be rescued within 6 hours
  3. Must be rescued within 12 hours
  4. Victims can remain on the spot

  These priority levels are associated with deadlines, the latest dates at which operations must be conducted at the demand points. Values for the priority coefficients and deadlines have been determined in collaboration with SDIS 31 and will be detailed in further chapters.

Priorities and deadlines are estimated by rescue teams according to the victim's situation, location, state of health, age, and other factors.

- **Categories**: Rescue teams must conduct various types of rescue operations and therefore use different categories of vehicles. These vehicles can handle different types of demands and also have different capacities. The problem considers victim relief operations where victims are rescued and driven to safety. So, the capacity limitations of vehicles are considered. The following vehicle categories are considered:

  1. Big size vehicles such as buses. This vehicle category conducts large-scale relief evacuations on low water levels. Vehicles of this category are often requisitioned by rescue teams from other services. These vehicles are not adapted to conduct operations when the water level rises. Other types of vehicles must be used in such cases.
  2. Specialized rescue teams vehicles. The vehicles of this category can conduct relief operations up to 60 centimeters of water maximum. Above these water levels, interventions are dangerous for the rescue team staff. When the water level is too high, a firefighter must walk in front of vehicles to make sure of the road path. But when the water is high this personnel can be endangered by sewer drains. The rescue capacity of these vehicles is limited compared to vehicles from the first category.
  3. When the water level is above 60 centimeters, boats are required for some operations. Boats also have limited capacity and are very slow. In practice, they are used to drive victims to a dry area and another vehicle is used to drive them to a safe place but in this work, a simplification is applied considering that victims are dropped directly at the rescue center.
  4. For emergency interventions or when a demand point cannot be accessed otherwise, helicopters can be used. They rescue one victim at a time but help to gain a lot of time, or can access very isolated locations.
  5. Finally, the last category has been defined for interventions needing specific vehicles for example for animals. For instance, during a flood in the Camargue region in the south of France in 2003, rescue teams had to rescue cattle. For such interventions, specific trucks are used. The specific interventions have been regrouped under this last category of resources.

- **No re-routing**: Rescue teams do not want to re-route their vehicles. Once a vehicle has started a route to serve a demand, its trajectory before completion of the service is not modified. This is for two main reasons. First, in a crisis management context, communication might be compromised so rescue teams prefer to avoid basing operations on communications. Then, rescue teams insist on the stability of the vehicles' routes.

For logistic reasons, they do not want to change route while a rescue operation is started. For instance, for locations that need to be relieved by boats, a truck drives the boats to the area where boats need to be put in water. If the vehicle is re-routed, the vehicle might already have reached the water, and firefighters are readying for intervention. Re-routing might then mean canceling the operation and its preparation.

- **Split service**: Rescue teams are often understaffed. It means that they do not have enough available rescue vehicles, or human power to drive them, to rescue all demands in one tour. As mentioned above, the capacity limitation of vehicles is considered. If the total number of victims to rescue is higher than the sum of vehicles' capacities, rescue teams need to practice *restocking*. This term is used to design the action of driving back to the rescue center (called *depot* in the VRP literature), to put victims to safety. Once a vehicle has done its restocking, it can start a new tour with its full rescuing capacity. Then, with that knowledge, it is permitted to serve demand with several vehicles. Some demands may have too many victims (quantity) to be served by one single vehicle.

- **Single depot**: For rescue teams, the routes are planned to start from the rescue center and also end there. Even when the place where victims are driven to safety is different from the location of the rescue center, rescue teams always go through the rescue center before starting a new tour. For this thesis, a single location is considered as the rescue center, vehicle depot, and safe place for victims.

Other characteristics of the problem have been defined in collaboration with rescue teams. These characteristics are necessary to build a VRP model:

- **Travel time**: To plan interventions in time, an estimation of the travel time is needed. It allows computation of dates of arrival at demand points knowing the date of departure from the previous intervention. The travel time is an estimation, computed according to the distance between consecutive demand points in a route and an approximation of the travel speed, common to all vehicles of a given category. Note that for the thesis, it has been considered that no demand point is isolated which means every location can be accessed from any other point of the territory, at least by a helicopter if there is no available road.

- **Service time**: For each rescue operations to conduct, rescue teams estimate a time to complete the action at the demand point. This time is called service time. The estimation is given by rescue teams according to the demand characteristics. It allows estimation of the end of service date at a node knowing the arrival date at this node.

1.2 Contributions

This thesis studies different solutions for the victim relief operations for flash floods and tries to answer the question: How to apply optimization techniques to improve decision making during flash floods victims relief operations?

The work of this thesis does not constitute a contribution to the operation research field but an application of optimization methods to the domain of crisis management where it has not been applied as in this thesis. To the best of our knowledge the crisis management problem of victim relief operations with demands priorities has not been studied under the model of a Vehicle Routing Problem.

1.2.1 Static contributions

A static version of the problem is studied first. This simplified variation of the real-life problem considers that all information is known at the beginning of rescue operations. A Mixed Integer Linear Programming (MILP) formulation is proposed to represent the static decision problem.

Solutions are developed and compared to solve the static formulation of the problem:

- An exact method: This type of method search for the optimal solution to an optimization problem and is able to prove optimality, but can be slow to compute feasible solutions for the VRP.
- Greedy heuristics: Heuristics that insert demands in the routes step by step. Past decisions are not questioned.
- Insertion heuristics based on the objective function and that question past decisions.
- Adaptation of a heuristic from the literature: Solomon's heuristic.

The solutions developed for the static variation of the problem are useful to tackle the dynamic version. Furthermore, the static formulation of the problem can be appropriate for training purposes for instance.

Floods often impact several places at the same time. Rescue teams are then confronted with the problem of resource dispatchment. Solutions are studied to optimally distribute vehicles between the different impacted sectors that rescue teams have to handle simultaneously.

1.2.2 Dynamic contributions

In a real-life VRP, especially in a crisis context, information is dynamic. During flood relief operations, all demands are not known at the beginning of interventions. Information about the state of the flooded territory and issues are revealed while rescue teams conduct rescues. This information may be revealed through:

- Scouting of rescue teams. As soon as rescue teams arrive in the impacted area, vehicles dedicated to scouting are sent to the territory to gather information. These vehicles send back reports of the flooded area and potential issues.
- Calls from victims to emergency numbers. During a crisis, a special call center is established at the rescue center to regroup calls concerning the flood. It gathers data about new demands that need to be served.
- Sensors deployed on the territory. In the i-Nondations project, for instance, the energy provider partner (Enedis) offers access to data from sensors placed on electric distribution infrastructures.
- Data collected from various sources. Research projects are presented in chapter 2 where data is collected with different approaches. For instance, data can be collected from satellite images or social networks.

Information may reveal new demands and updated data. Updates may be about demands or the road network. This dynamic information may also be encountered by the rescue vehicles while conducting operations. A road can be blocked for instance. All this dynamic information must be integrated into the problem and the solutions must take it into account when planning routes. Solutions are developed in this thesis with two main approaches:

- A re-optimization approach adapting solutions from the static context.
- A dynamic insertion heuristic.

Routes computation strategies have been developed to determine when the solution is updated with new information. Three strategies are presented and evaluated in the thesis.

Furthermore, metrics to measure the characteristics of evaluation scenarios are studied. Two metrics are observed:

- **Dynamism**: A metric from the literature that measures the distribution of dynamic demands on the time horizon of the crisis.
- **Quantity distribution**: A metric proposed in this thesis to evaluate the distribution of the total number of victims rescued during the crisis over the different demand points.

The influence of these metrics on solutions performances is observed.

1.2.3 Evaluation tools contributions

For evaluation purposes, different softwares have been developed in this thesis. First, a graph generator has been developed. Graphs are used to represent a flooded territory through its issues (vertices) and road network (edges). The software can create graphs for all types of flooded territory, from a countryside

area to a dense city. It has been used to create graphs with similar characteristics to the Luchon flood from 2013. This flash flood happened in the south of France and is one of the cases studied in the i-Nondations project.

For evaluation of real-life dynamic crisis scenarios, a simulator is presented. This software runs crisis scenarios in simulated time and represents each rescue team entity with a different process. Then, crisis scenarios similar to the case study can be run to validate the performances of the solutions proposed in the thesis.

1.3 Plan of the manuscript

Chapter 2 reviews the related work in both research fields of crisis management and VRP.

Chapter 3 presents the tools developed for evaluation purposes, the graph generator and the simulator.

Chapter 4 presents the static problem along with solutions and their evaluation.

Chapter 5 presents the dynamic version of the problem, developed solutions to answer it and their comparison.

Finally a conclusion of the work is presented in Chapter 6.

1.4 Ethical discussion

In this thesis, victim relief operations are studied. As mentioned earlier in the introduction, priorities are applied to demands and therefore to victims. This subject is ethically sensible and therefore it felt important to discuss it.

The priority is the strict application of a metric used by rescue teams from SDSI 31 during their operations. It translates through different factors the ability of a victim to wait before being rescued. The priority aims at reducing the discomfort of victims and not at selecting victims to be rescued or not. Even with the degraded state, victims who are put in lower priority categories are considered to be in good enough conditions to wait for rescue. The objective of rescue teams is to reduce the chances of a situation getting worse for victims.

In addition, the work presented in this thesis aims at being a support decision tool. Routes for rescue vehicles need to be validated by experimented crisis managers and the developed solutions are meant to accelerate the decision process or to give keys to improvement in the response to flash floods but they should not be used without proper supervision nor should they replace the expertise of trained personnel.

To conclude on the ethical aspect of my research, the carbon footprint of my thesis has been evaluated. It has been calculated using green-algorithms.org v2.0 [59]. The calculation considers the large-scale experimentation conducted on the platform Grid5000 located in France. Experimentation took a total of

576 hours of computations on 16 CPUs Xeon E5-2660 v3 and has drawn 207.47 kWh. It represents a carbon footprint of $8.08kg\ CO_{2e}$.

Chapter 2

State of the Art

The work in this thesis is to optimize the rescue teams' response to flash floods. The problem of this thesis is a VRP where the objective is to optimize the routes of rescue vehicles. A large literature exists on this subject and its different variations. The thesis is also concerned with the disaster management literature. Different crisis management problems are regrouped under that field of research and various types of solutions are proposed. The purpose of this chapter is to position the studied problem regarding both scientific domains. The analysis of both fields is done separately.

2.1 Disaster management

Disastrous events result in a lot of damage and casualties in the world every year. There are different types of disastrous events. A disastrous event is either from natural or human causes. Natural disasters are hurricanes, fires, floods, earthquakes, typhoons, storms, ... Disasters might also be caused by human interventions such as nuclear incidents, toxic leaks, ... The crisis management literature studies solutions to reduce the impact of such events. Each type of disastrous event is very different in the response. Furthermore, the action to tackle the problem can be made at different levels. In crisis management, 4 phases are considered, as mentioned in [88]:

- Mitigation: It consists in trying to avoid the crisis or reduce its impact if the crisis cannot be stopped. This phase includes engineering solutions, preventing development (of households for instance) in hazardous areas, infrastructures protection, natural environment protection to serve as a buffer, education of the people about the risk, and adapted insurance development.

- Preparedness: It regroups all the activities that help improve response immediately after an incident. It includes the development of response

procedures, warning systems, evacuation planning, exercises to test emergency operations, and training of emergency teams.

- Response: Gathers activities during the crisis or immediately after a disaster to meet the urgent needs of victims. It includes all emergency operations.
- Recovery: Actions conducted after the crisis end and when the urgent needs have been met. It includes infrastructures repair and restoration as well as all operations designed to put the impacted community back together.

These stages are considered cyclic in the literature as illustrated in fig. 2.1.

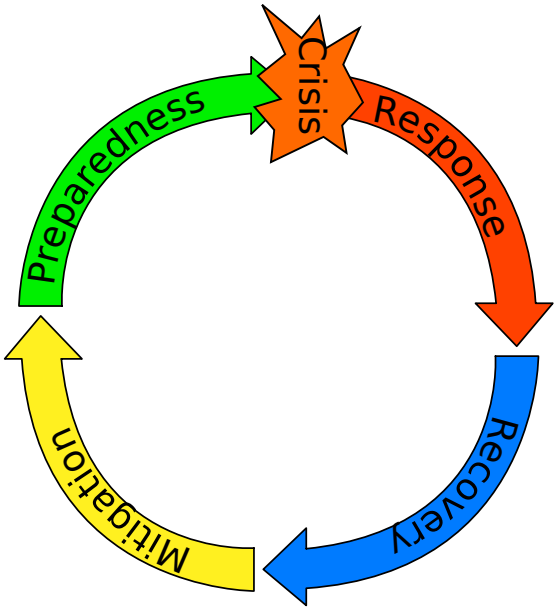


Figure 2.1: Crisis management cycle

The work presented in this thesis is focused on a case study in France. Different institutional databases are available for the disaster management actors in France:

- **BD TOPO**: Three-dimensional vectors database covering France. This database is updated every year by the National Institute of Geographic and Forest Information. This database contains:
  - The road network
  - The railway network
  - Energy transportation network

  - Buildings
  - Points of activity or interest
  - Named locations (localities, orography, protected natural areas ...)

- **MAJIC**: National database on land data. It is used for taxation purposes and managed by the CEREMA (partner in the i-Nondations project). It is updated every year and lists information on the typology of buildings such as their age, size, state, and occupation status. It also covers housing infrastructures.

- **GeoSirene**: Database based on **Sirene** database which adds it geolocation data. **Sirene** describes all companies and establishments in France. It is updated daily and managed by the National Institute of Statistics and Economic Studies (INSEE). This database completes the previous ones with the following information:

  - Company addresses
  - Number of employees of companies

- **RPG**: Geographical database used by the French Ministry of Agriculture. Data is aggregated at the scale of an agricultural parcel. It gives information about the culture of each parcel declared by its farmer.

- **DDT Maps of Potential Flooded Area**: The Departmental Direction of Territories (DDT) for each department maintains maps of the potentially flooded areas. It contains:

  - A table summering the main issues detected (affected population estimation, inhabited buildings, Buildings Hosting Public, ...)
  - The materialization of these issues on ZIP maps and Google Earth (issues and estimated water high)

These databases are an important part of the information system. It is very useful for operations of the response phase. Data is also used in the preparedness phase with the example of VIGInond we will detail below. They are used in the response phase to collect data about the situation on the field to plan rescue operations. For instance, knowing which enterprise is in business and how many people it hosts helps rescue teams to scale the response if the crisis happens during the day. The same example can be transferred to households for crises happening outside of working hours. Works based on big data use some of these databases.

In the following literature review, the different presented approaches are regrouped by phase.

2.1.1 Mitigation

Very various works in the literature study the Mitigation phase of crisis management. For instance, [77] studies the resilience of water distribution systems to different types of hazards. Water distribution is vital, however, it can be disturbed by different hazards. This Franco-German project studies ways to detect abnormal events impacting the distributed water quality and integrity through monitoring and vulnerability analysis methods. The goal is to avoid distributing contaminated water and therefore escape a poisoning crisis on a large scale.

The paper [27] reviews the risk management in France with a focus on the Mediterranean area. It also offers policies in collaboration with companies and public institutions to mitigate the impact of floods. It evaluates these preventive policies and the impact of prevention to test if it helps improve the response.

In [20] 752 households prone to flood are studied. It allows showing the influence of coping appraisal on flood mitigation. A coping appraisal is an individual adaptation to face a hazard, flood in this case. By evaluating individual measures put in place in households vulnerable to floods, one can get insight into which measure is the most efficient to reduce floods' impact.

The same type of study is conducted in [79]. In this article, 885 french households are studied with a protection motivation theory approach. These articles conclude that coping appraisals are more effective than threat appraisals to mitigate the impact on households. It also gives recommendations to improve flood preparedness.

The article [26] presents two methods to assess the exposure of daily commuters to flood risk. This work allows determining critical road intersections that need to be adapted to reduce the impact of a flood on the road infrastructure. The evaluation of the geometry helps to limit victims among commuters by avoiding damages to the infrastructure they use. It also helps rescue teams during rescue operations since the road network helps to speed up the evacuation process if it is not damaged.

2.1.2 Preparedness

This phase can regroup various types of approaches to tackle a crisis. For instance, [21] studies a warning system for floods. In this article, a method is developed to monitor the activity of a river. The studied river is the Tagliamento in Italy. A neural network is developed to predict the water level at different points of the river in the knowledge of distributed raining information upstream of the river. The purpose is to detect dangerous water level predictions and forecast them.

A review of the different measures that can be put in place is given in [58]. It is focused on flood risk, more specifically on land-use planning and private precautionary adaptations. It details measures proven the most effective in several case studies.

During the preparation phase, rescue teams have data to prepare plans for

interventions and also to detect a potential crisis as early as possible. For floods, in France, diverse monitoring systems exist. Vigicrues is a national information service on flood risk that gathers various data to affect a vigilance index over areas close to the main rivers on the territory. Authors of [35] study data collection and interpretation to enrich data from Vigicrues with another system VIGInond. The purpose of this work is to develop a new version of Vigicrues coupled with VIGInond which was only available for interior affairs ministry use.

The work from [30] also studies flash floods. The goal of this work is to detect flash flood hazards in selected agricultural regions using satellite data. This data is coupled with models that allow prediction of the level of hazard to the region is submitted. Then, authorities can adapt the alert system according to the measured index as it is specified in the article. It may also be used to test the system in place in a given area with the help of simulations.

2.1.3 Response

The article [50] is at the limit between preparedness and response phases. A variable selection technique is offered to help the decision and prediction of firefighters' interventions.

The article [36] conducted in the GéNéPi project works on big data to improve the number of usable data sources. This process allows getting more information so that better decisions are made in the response phase. The goal is to get a better understanding of the on-field situation thanks to big data. It helps to feed rescue teams with more knowledge in the decision-making process.

The work from [14] follows the same type of approach to improve response. In this article, tools based on big data are presented and tested in real-case scenarios.

In [87] radar detection is used to compute a 3D model of the flood. It allows for measuring the impact of the flood and therefore the water level in all the impacted areas. This type of detection system could be of great use for rescue teams to adapt the resources to the local severity of the flood.

A different approach is offered in [72] and [73]. Both articles develop an approach with meta-models to tackle the decision support issue for crisis management. This approach creates a diagram for each type of crisis. These diagrams are created with a modeling language similar to UML (Unified Modeling Language) for instance. However, in this case, it models the events that can occur during the crisis and the appropriate associated decision. These meta-models help rescue teams to gain time in the decision-making process. It helps to follow clear procedures to improve response to different types of crises.

The study [23] presents a multistage stochastic program for urban search and rescue operations. Search and rescue is an important part of the response phase and can be applied to many types of disasters such as earthquakes for instance. In such a crisis, the impacted area is devastated and the objective of the developed solutions is to optimally deploy the rescue teams to find victims.

As mentioned in the introduction, the problem studied in this thesis can be categorized as a VRP. The victim relief operations are part of the response phase. Different works have been led on VRPs for crisis management problems. For instance, [18] studies the use of a Genetic Algorithm to support the decision for transportation in disaster relief operations. Contrary to our problem, this article focuses on the delivery of humanitarian goods and equipment to defined locations.

The work from [102] studies a Pick-up and Delivery Problem (PDP) for supply distribution in disaster relief operations. A PDP is a VRP where there are pick-ups as well as deliveries. In this problem, due to the crisis management context, travel times may vary. Therefore, to solve the less-than-truckload freight problem, stochastic programming is used.

In the article [56], the work focuses on the dispatching of emergency vehicles and applies its study to earthquakes. In this article, data fusion is used. Here, precise locations are not considered. The objective is to re-locate and route vehicles where they are most likely to be in use to rescue victims according to the data fusion process. Then when emergency vehicles have rescued victims, they are routed to the appropriate hospital. In the data fusion process, different values are available such as the priorities of victims. The goal is to generate routes in the knowledge of the field situation (road network state, congestion ...). The paper [91] proposes a fuzzy-clustering optimization approach for relief supplies distribution operations. The objective of this study is to aggregate demands in clusters and route distribution vehicles to these clusters.

The distribution of goods for disaster victims is also studied in [80]. In this paper, local distribution facilities are created and the studied problem is a last-mile distribution problem. Authors of [74] also study a last-mile distribution VRP for relief operations after an earthquake.

In [85], the studied problems are a road network accessibility problem and a work-troops scheduling problem. The first one studies how to find usable paths for rescue vehicles in the aftermath of an earthquake. After such a crisis, road segments are often blocked by rubble, and finding a practicable path to a rescue point might be a real challenge. The second problem studies how to generate a schedule to access these areas. The article evaluates the developed solution to Haiti's earthquake in 2010. The evaluation graph contains more than 10.000 vertices and edges.

Finally, the paper [2] also studies evacuations applied to floods. However, it considers evacuations with the assumption that people use their vehicles and are not evacuated by rescue teams' vehicles. In this context, the objective is to plan evacuation routes to forecast the victims to ease the evacuation process and avoid congestion. This type of approach is more adapted to general floods but less effective in flash floods context since the preparation period is reduced strongly.

As it has been developed in this section, several works have been conducted in the response phase of a crisis. Various problems are studied in the literature and different approaches are tested to solve these problems. However, to the best of our knowledge, no study has been conducted on the victim relief operations

of a flood considering the problem of scheduling routes for rescue vehicles as a VRP. Furthermore, the problem we study in this thesis has specific constraints due to flood crises and the rescue teams' operations process from Firefighters.

2.1.4 Recovery

The article [90] gives insights to evaluate the post-crisis benefit of flood-proof improvements on the economic impact of a flood crisis. This study helps prioritize the arrangements to apply in the build-back process during the recovery phase.

In [93], an emergency response service is presented. It aims at giving reliable updated information for the post-crisis phase. This service uses sensors data coupled with satellite images analysis to give reliable information on the evolution of the situation in the field.

Authors of [83] develop indicators to evaluate the social vulnerability post-crisis. 67 case studies are analyzed to develop indicators that reflect the context in which social vulnerability develops.

On another aspect, [57] focuses on the "build back better" question. The study of this article is led on building back the infrastructure on an impacted territory to improve its resilience to a potential future flood.

In this section different problems in the crisis management literature have been presented. These problems have been put in the perspective of the phase of the disaster management they intervene in. The problem studied in this thesis is part of the response phase. Various types of problems studying this phase have been reviewed and it reveals that the problem we study here does not find its equivalent in the crisis management literature. However, as mentioned in chapter 1, the studied problem is also a VRP. Various literature exists on that problem and the next section focuses on related work on VRP.

2.2 Vehicle Routing Problem

The Traveling Salesman Problem (TSP) is a NP-hard problem of optimization where a salesman has to visit a list of customers exactly once. The objective is to optimize the order of visits to all the customers in terms of route cost, given that every road between two customers is associated with a route cost. This problem was introduced by Hassler Whitney in 1934 [39]. The NP-hard complexity of the problem induces that there is not algorithm to solve this class of problem optimally in polynomial time – unless $P = NP$.
The Bin-packing problem is also a class of NP-hard problems where the objective is to optimally arrange elements of various sizes inside a container of limited capacity [32].
The VRP is a combination of these problems introduced by [25] in 1959. In a VRP, the objective is to optimize route cost, like in a TSP, as well as optimizing the load of the vehicles like in a bin packing problem. In a VRP, the purpose is

to serve several locations usually called customers with one or several vehicles from a single depot. Every customer must be visited at least once during the time horizon of the problem and the routes might be constituted of several tours between customers and the depot.

This problem is also NP-hard as confirmed by [65]. On top of the general VRP, various other constraints might be added leading to variations in order to fit every problem's constraints. In the following of this chapter, the literature of the VRP is overviewed through its most widespread variations.

2.2.1 VRP variations

Different variations of the VRP have been introduced through researches in the field. In this section, a non-exhaustive presentation is made of the major VRP variations. In the thesis [12], a similar state of the art organization is offered.

Capacitated Vehicle Routing Problem

This is the most common variation for a VRP. In a taxonomic review, [33] where 30 articles have been studied, only two of them do not consider capacity. In the Capacitated Vehicle Routing Problem (CVRP) the capacity limitations of the vehicles are considered. Vehicles cannot handle more than their capacity.

A model for the CVRP is presented. This model has been offered by [38] and [37] as a MILP. A MILP is a formulation in which the problem is described by an objective function, linear constraints, and integer variables. It is based on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_0, \ldots, v_{n-1}\}$ is the vertex set of size n where vertices (nodes of the graph) represent the customers to be served except for v_0, the node of the depot. $\mathcal{E} = \{(v_i, v_j) | v_i, v_j \in \mathcal{V}, i \neq j\}$ is the edge set.

Let us consider:

- $Z = \{0, \ldots, n-1\}$ is the set of indexes of vertices in the graph and $Z^\star = Z \setminus \{0\}$
- $M = \{1, \ldots, m\}$ the set of available vehicles
- Q the maximum capacity, constant and common to all vehicles
- q_i the capacity (number of items or victims) to collect at node of index i
- $d_{i,j}$ the distance between nodes of indexes i and j

And the decision variables:

- y_i^k a binary variable equal to 1 if node of index i is visited by vehicle k and equal to 0 otherwise
- $x_{i,j}^k$ a binary variable equal to 1 if edge (v_i,v_j) in the route of vehicle k and equal to 0 otherwise

The CVRP is then modeled as follows:

$$\min \left(\sum_{i \in Z} \sum_{j \in Z} \sum_{k \in M} x_{i,j}^k \cdot d_{i,j} \right) \tag{2.1}$$

$$\forall k \in M, \quad \sum_{i \in Z^\star} q_i \cdot y_i^k \leq Q \tag{2.2}$$

$$\forall i \in Z^\star, \quad \sum_{k \in M} y_i^k = 1 \tag{2.3}$$

$$\sum_{k \in M} y_0^k = m \tag{2.4}$$

$$\forall j \in Z, \ \forall k \in M, \quad \sum_{i \in Z} x_{i,j}^k = y_j^k \tag{2.5}$$

$$\forall i \in Z, \ \forall k \in M, \quad \sum_{j \in Z} x_{i,j}^k = y_i^k \tag{2.6}$$

$$\forall k \in M, \ \forall X \subset Z \text{ s.t. } 2 \leq |X| \leq n-2, \quad \sum_{(i,j) \in X^2} x_{i,j}^k \leq |X| - 1 \tag{2.7}$$

This mathematical model begins with the objective function in eq. (2.1) that determines that we want to minimize the total distance traveled by vehicles. Constraints in eq. (2.2) express the maximum capacity of vehicles. They ensure that the sum of served quantities at each node by a vehicle does not exceed its capacity. These constraints could be removed if the problem is not a CVRP but a general VRP. In constraints of eq. (2.3), the visit limitation for each node is expressed. It ensures that a node is visited only by a single-vehicle. In this formulation, exactly one vehicle goes through each demand and can serve it only once, therefore any problem in which a quantity would be bigger than the maximum capacity would be infeasible. Equation (2.4) rejects solutions where each vehicle does not visit the depot. Constraints eq. (2.5) and eq. (2.6) force solutions to travel to a node with a vehicle if the node is served by this vehicle and to depart from it after service. Finally, constraints eq. (2.7) allow the elimination of solutions that do not visit all customers because of sub-tours. It states, for each vehicle, that for every sub-set of nodes of Z, the number of visited nodes is lower or equal to the number of nodes in the sub-set. Therefore this constraint eliminates a solution where a vehicle would visit a node twice and consequently leave a node not served.

This model is general and can be applied to most CVRP. Constraints would need to be added for other variations in order to complete the model though.

Vehicle Routing Problem with Time Windows

The Vehicle Routing Problem with Time Windows (VRPTW) has been introduced in [46] and developed in [92]. In this problem, a time window is associated with the demands. The time windows are described by a lower bound and an upper bound $[l_i, u_i]$. In this problem, the demands have to be served within their time window, *i.e.* after the beginning of their time window and before its end. In the literature, the time window problems consider the constraint from eq. (2.8) in various ways. In commercial problems mostly, the time window can be violated under a penalty. It can also be a strong constraint where a solution that violates a time window is not valid.

The following constraints could be added to a VRP model to turn it into a VRPTW. These constraints reflect the case of a strong time window. A decision variable is added to the model. h_i^k is the date of arrival of vehicle k at node i. If the node is not visited by the vehicle, the date is equal to 0.

$$l_i \leq \sum_{k \in M} h_i^k \leq u_i \quad \forall i \in Z \tag{2.8}$$

The constraints in eq. (2.8) make sure that the date of arrival at a node is within the time window of the demand. The decision variable can be computed using an approximated mean travel speed for the vehicles. Knowing the distance that has to be traveled, the estimated date of arrival can be computed.

Heterogeneous Fleet Vehicle Routing Problem

The Heterogeneous Fleet VRP considers vehicles with different vehicle capacities. It has been introduced in [47]. Contrary to the model presented above, the capacity is not common to all vehicles and is not constant. Therefore, in this problem, the maximum capacity of vehicle k is noted Q_k. Furthermore, the maximum capacity constraints eq. (2.2) is also modified for such problems:

$$\forall k \in M, \quad \sum_{i=1}^{n} q_i \cdot y_i^k \leq Q_k \tag{2.9}$$

The difference is that constraints ensure that each vehicle does not exceed its own capacity.

Split Delivery Vehicle Routing Problem

Split Delivery VRP has been presented first in [31]. In this variation, it is considered that the customer can be served by several vehicles instead of only one in the CVRP presented model. It might be necessary in cases of big demands that cannot be served by only one vehicle because of capacity limitations. For the Split Delivery VRP, the constraints in eq. (2.3) are removed from the model since they ensure a customer is served only once. The variable q_i is not used and q_i^k is used instead which is the quantity served at a node i by a vehicle k.

Periodic Vehicle Routing Problem

The Periodic VRP has been introduced with [13]. It has been later formally defined and put into equations in [84]. In this variation, the service to the customer has to be performed over several periods. This is similar to the VRPTW but instead of having one time Window, several time windows (periods) are considered. The set of periods for the customer i can be noted $P_i = \{[l_i^1, u_i^1], \ldots, [l_i^{n^i}, u_i^{n^i}]\}$ with n^i the number of periods of service for node i.

We consider the model of the general VRP (without the constraints in eq. (2.2)), and the decision variable h_i^k that becomes $h_i^{k,r}$ with $r \in [\![1, n_i]\!]$ the date of service of node i by vehicle k on period r (this value is null if the node is not served by vehicle k on period r). Then the model needs to be enriched with:

$$\forall i \in Z^\star,\ \forall k \in M,\ \forall r \in [\![1, n^i]\!], \quad l_i^r \leq \sum_{k \in M} h_i^{k,r} \leq u_i^r \tag{2.10}$$

The constraints eq. (2.10) adapted from eq. (2.8) state that if the node i is not visited by vehicle k, all members of the equation are null and the constraint is true. Else, the constraint is similar to those from eq. (2.8). In addition to these constraints, the constraints defined by eq. (2.3) should be removed from the model. In fact, each node can be visited more than once if it has several visit periods.

Pick-up and Delivery Problem

The PDP as defined in [86] is a VRP where people and goods can be picked up as well as delivered to customers' locations. The main challenge in this problem is to be able to deal with the load of the vehicle and deliver it before a pick-up for which there is not enough space in the vehicle.

Dynamic Vehicle Routing Problem

In the Dynamic Vehicle Routing Problem (DVRP) information is considered dynamic in opposition to the general VRP where it is static. In a dynamic problem, part of the information about the crisis is released dynamically. [81] lists three majors sources of dynamism in DVRPs.

- Dynamic customers: It describes problems where the points of demands (location and characteristics) are revealed dynamically. For example, [43] studies the relocation and deployment of ambulances with interventions that are 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. For instance, [44] studies the PDP with dynamic requests and updates solutions dynamically with the use of a neighborhood search heuristic.

- Dynamic travel times: In these dynamic problems, the travel times of the vehicles are considered to be variable over time like in ([100]). This article studies the TSP where the travel time can vary with the date of departure.

Articles such as [17] or [76] offer reviews on the literature of DVRPs. In order to measure how dynamic a problem is, metrics have been defined and commonly used in the literature. The most widespread is the Effective Degree of Dynamism developed in [63] and [64]. Another metric called Dynamism offers a different approach [98]. This metric will be discussed in chapter 5. In terms of model adaptation, a release date must be added, r_i, which is the date dynamic information about a node i is revealed. Then the model may re-use the VRPTW model with the beginning of the time window being the release date. Such a model would be adapted to a dynamic customer VRP.

Stochastic Vehicle Routing Problem

This variation is very similar to the DVRP but probabilistic information is available about the dynamic factors. The difference is highlighted because it opens a different range of solutions using these probabilities about future demands. For example, [97] studies a stochastic problem where information about the customer are only revealed upon arrival at the node. However probabilistic information is known in advance about the size of the demand to pick up.

The major variations of the VRP have been presented and all that is related to our problem, however there exists a lot of other variations in the literature of VRPs such as the Dial a Ride problem [24] for instance.

The following state of the art focuses on the approaches from the literature to solve such problems.

Note that for the rest of the manuscript, the following terms are used to be more adapted to a crisis management vocabulary:

- **Customers** is replaced by **Demands**. The term to describe customer locations are diverse:
  - Node
  - Demand point
  - Issue
- The word **request** is used only to describe the characteristics of a **Demand**.

2.2.2 Resolution methods for the Vehicle Routing Problem

Depending on the characteristics of a VRP, different resolution methods can be used. In this section, we present the main approaches that are studied in the literature.

Exact methods

When one wants to optimize vehicle routes for a VRP, exact methods can be used. An exact method ensures to find an optimal solution. Solutions that are not evaluated are proven not to be optimal solutions. VRPs models are often defined as MILP. A MILP describes a problem with a cost function, linear constraints and integer variables. An example of a MILP model is given above in section 2.2.1.

MILP problems are generally solved with a linear-programming-based Branch-and-Bound algorithm. This algorithm was first developed by Ailsa Land and Alison Doig in 1960. The name "Branch-and-Bound" first appeared in [66]. It refers to the structure of the algorithm as a research tree. In this tree, nodes represent a sub-space of the search area, and branches are created to divide the search space. When the optimal solution of sub-space has been found, there is no need to divide the search space and branch. This solution is a leaf from the search tree. Let us consider a problem where we want to find a value of an x, that takes its values in a search space S, such that their image under an objective function f. In other words, we look for an element of: $\arg\min_{x \in S} f(x)$. The basic principle of a Branch-and-Bound algorithm can be summarized in two main steps:

- **Branch**: Recursively splitting the search space into smaller spaces. Then trying to minimize $f(x)$ in these sub-spaces.
- **Bound**: Using only branching would be equivalent to a brute force algorithm where every solution is tested. To improve the search for the optimal solution, the algorithm records bounds for the minimum. These bounds are used to prune the search space and eliminate branches that are guaranteed not to contain an optimal solution.

At each step and for each sub-problem, integrality restrictions are removed. The resulting Linear-Programming relaxation can then be solved. If the result of the Linear-Programming relaxation satisfies all the integrality restrictions, it is recorded as a feasible solution. Else, a branching is executed on the branching variable. For instance, with the example of x and $f(x)$ mentioned above, if the result of the LP relaxation gave the value $x = 2.3$, and this solution does not respect the integrality restrictions, branching creates two new nodes: $x \leq 2$ and $x \geq 3$.

As illustrated in fig. 2.2, two sub-problems P_1 and P_2 are created from P.

If an optimal solution is found to P_1 and P_2, the best of both solutions can be taken as the optimal solution of P. Then, a search tree is created by branching recursively. The objective is to find the optimal solution to the main problem by resolving sub-problems optimally. During the search in the tree, when a feasible solution is found at a node, it is registered and moved up in the tree. If the objective value for this solution is better than the upper bound, it replaces it. These solutions are called incumbents. Then during the execution of the algorithm, we will not accept an integer solution with an objective value higher

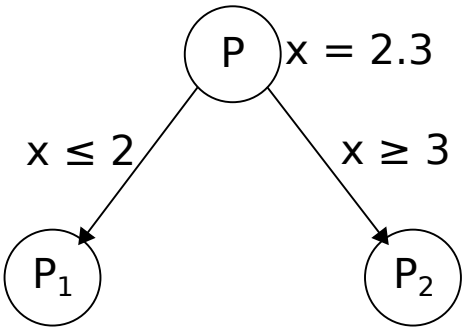


Figure 2.2: Example of the branching for a LP relaxation's solution x = 2.3

than this incumbent. The algorithm also uses a lower bound often called the best bound. This bound is computed as the minimum of all optimal objective values of all leaves in the search tree. The difference between the Upper and Lower bounds is called the gap. When the gap reaches 0, the optimality of the solution is proven. The value of the gap can be adjusted in most solvers to define the desired accuracy of the result.

The branch-and-bound algorithm can be coupled with cutting planes methods. The combination of cutting planes and branch-and-bound is called Branch-and-Cut. Cutting planes tighten the formulation by removing undesirable fractional solutions. The cutting plane algorithm intervenes when an optimal solution is found to a sub-problem of the relaxed problem. If this solution has a non-integer value, the algorithm allows adding constraints that do not exclude any integer solution but exclude some of the non-integer solutions. The principle is illustrated in fig. 2.3. In this figure, the search space is illustrated by the grey area, whose bounds are determined by the constraints of both the main problem and the constraints that have been added during the search in the current part of the search tree. Feasible solutions to the problem are represented by the black dots. As we can see, since this example takes place in a sub-problem of the main problem we are trying to solve, the objective direction goes out of the search space (orange arrow). The global optimum of the problem is outside the search space of the sub-problem. In this example we are trying to find values of variables x and y that minimize $f(x, y)$. One can imagine a three-dimensional plot, where the third z-axis is added, and represent a surface defined by $z = f(x, y)$. The objective is to find the lowest point of the surface (along the z-axis). The red dot is a solution found for the LP relaxation. This optimum of the sub-problem contains a non-integer value. It can be verified in the figure, where $x = 3$ and $y = 2.5$ for the red dot. The cutting plane algorithm adds constraints to the problem to eliminate the optimal solution with non-integer values. Here are displayed the constraints $x+y = 5$ and $y-x = -1$. They allow to eliminate the solution with non-integer values and select the LP optimum (green dot).

The main drawback is that the exact methods' computation time increases

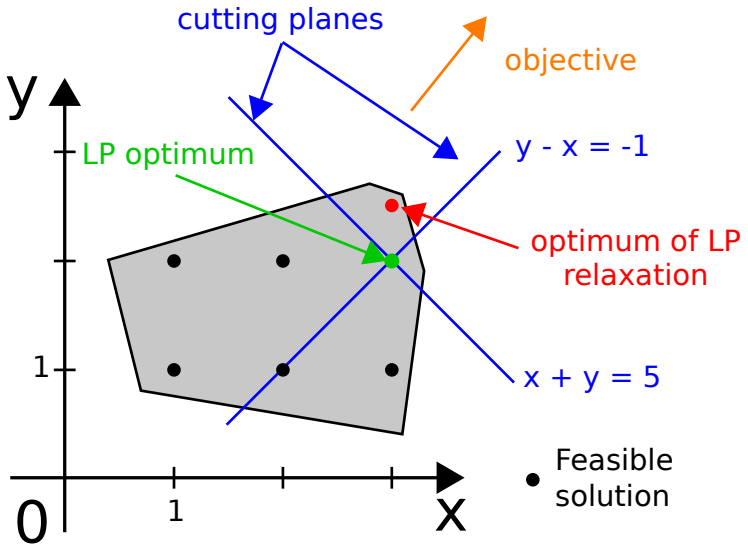


Figure 2.3: Example of step of the cutting plane algorithm

exponentially with the scale of the problem. When solving an NP-hard problem, the number of iterations of the algorithm is exponential with the number of nodes in the problem. Hence it cannot be an appropriate approach when computation time must be kept low.

Literature reviews such as [60] offer a survey of exact methods to solve VRPs.

The work presented in [82] offers to solve a Periodic VRP using exact methods for instance. In addition to the classic Periodic VRP, customers must be served by the same driver in this problem. The resolution method is a branch-and-cut algorithm. In this article, the planning horizon is of several days, and it is acceptable that solving runs for several hours.

The article [7] also studies the Periodic VRP however the problem is relaxed with flexibility hypotheses: the quantity to deliver at each period is not fixed. In this work, a total quantity has to be delivered to each customer and a maximal quantity is fixed for each periodic delivery. The paper shows that relaxing the general Periodic VRP allows to improve routing costs. It is also solved using an exact method and valid inequalities are presented for the problem.

Exact methods can also be used to solve stochastic problems. For instance, [95] solves it with CPLEX ([53]) (one of the most widespread commercial solvers) considering uncertainty over the worst-case scenario. When it is possible, this method allows transposing a Stochastic VRP into a general VRP and to solve it with exact methods.

Other approaches allow using exact methods for stochastic problems. The paper [41] deals with a problem where customers are uncertain as well as the quantity of the demands. In this pick-up problem, an exact method is used and coupled with policies that state that a customer is skipped if it is revealed

absent. A penalty is applied if the vehicle is full and restocking is forced.

However exact methods are limited by their computation time performances and when real-time problems are studied, for instance, the use of heuristics may be more appropriate so that we can find a solution where all constraints are fulfilled, even if it may not be optimal in regard of the objective function.

Heuristics

A Heuristic algorithm solves an optimization problem without an optimality guarantee but within a reasonable computation time. They have been used very early to cope with the computation time drawbacks of exact methods. The best heuristic example is greedy algorithms. This type of algorithm is meant to be very simple and fast. It builds the solution by trying at each step to achieve a locally optimal choice without questioning past decisions.

Early on, [47] used a heuristic solution to help to improve the performance of an exact method for a problem of Heterogeneous VRP. In this problem, finding the optimal fleet size is also studied.

In [92], the author has developed insertion heuristics and tested them on benchmarks that are still used as a reference. Benchmarks are sets of graphs on which the evaluation is conducted. The optimal value of the objective function is often given as well. They are generated depending on various parameters (like the number of nodes in the problem or the total quantity to serve for instance) and allow comparison of results on the same set of graphs as other studies. The heuristics are references in the literature. The principle of insertion heuristics is to build routes by inserting demands iteratively into the existing routes. The objective is to optimally decide on the vehicle and the position of insertion. Contrary to greedy algorithms, by inserting demands at different positions and not only at the end of the route, previous insertions, and therefore previous decisions are questioned.

Studies such as [69] or [8] offer a double horizon heuristic. The principle of the rolling horizon is to consider different time periods, in these cases too. In the short-term period, a feasible solution must be found quickly for the first demands to be served. This solution needs to be computed quickly to start serving customers, at the expense of solution quality. In the long-term horizon, while customers of the beginning of the road are being served, another heuristic tries to improve the solution. In the work [101], the same heuristic is applied in three phases. These heuristics are adapted for problems where the dynamism is low and demands that are being served or about to be served are not questioned by more urgent dynamic demands.

Heuristics can also be used to solve stochastic problems. In [104], a heuristic is developed to generate several routes. A route is computed for every possible value of an uncertain variable and the most probable route is followed until uncertain information is realized, *i.e.* becomes certain. Heuristics are of great use with such approaches because they allow the computation of feasible solutions very quickly and to adapt to new releases. The same approach is followed by [51] with a problem where the capacity limitations of vehicles are not considered.

The multiple scenarios approach is also studied by [15] and [16] on stochastic problems. Strategies are presented to determine when to delete routes in the pool of computed routes.

Finally, [23] also uses a two-stage approach with stochastic requests. In this article, two related stochastic programs compute routes on different time horizons, short and long term.

Meta-heuristics

A Meta-heuristic is a heuristic that tries to cope with the main default of heuristic that can get stuck in a local optimum during the research of solutions. In a meta-heuristic, a mechanism that tries to find the local optimum is coupled with a diversification method that variates the initial space for the search for solutions. The objective is to explore the solution space the most effectively possible to avoid being stuck in local minima and to focus quickly on promising regions of the search space.

One of the most widespread meta-heuristic is called Tabu Search and was introduced by [45]. It operates on a list of Tabu that records the last visited solutions to avoid coming back on solutions that have already been tested. The algorithm then avoids solutions from the Tabu list when searching for a better solution in its neighborhood.

In [43] this meta-heuristic is used to pre-compute redeployment scenarios for an ambulance dispatchment problem. In this article, every time an intervention is needed, the closest ambulance is sent. Therefore the rest of the fleet, on standby, might need to be re-dispatched to optimally cover the area.

The work from [102] solves the initial instance (at the beginning of the problem, a first instance with the static demands is available) using a Branch-and-Cut algorithm with time-dependent travel times. Stochastic modeling is used to handle uncertainty. Then to deal with the dynamic customers in the problem, a tabu search heuristic is applied to find a new solution with the added dynamic customer. The same principle is applied in [44] to find feasible solutions dynamically. The presented solution offers computation times of several seconds for the insertion of a single demand and the improvement of the solution. The overall computation time is counted in minutes to hours.

Tabu Search has been used in [42] in a parallel system to increase computational performances. This solution can be parallelized and the different processes can look in different search spaces, hence avoiding duplicated work. The Tabu list may be shared by the parallel processes to avoid converging on the same local optimum. The problem studied in this work is a courier distribution problem where capacity limitations are not considered. [54] also applied parallelism to a Tabu search heuristic. In this article, the dynamic customers are integrated into the solution as above with a threshold-based waiting process to leave time to compute updated solutions.

Authors in [61] and later [62] present a Variable Mixed Integer Programming Neighborhood Descent algorithm (VMND). In this algorithm, a branch-and-bound algorithm is used and the VMND is embedded inside it. It is a local search

algorithm that tries to improve the solution by searching for better solutions in a neighborhood of the new incumbent when such an incumbent pops. The general Variable Neighborhood Search (VNS) heuristics and its variation VMND avoid being stuck in a local minimum. A new family of inequalities is presented to improve the performance of the branch-and-bound algorithm. The problems studied in these articles are still over several days' horizons. The computation time is around an hour or more for instances of the scale we want to study in [62] for instance.

Another type of meta-heuristic commonly used is Genetic Algorithms (GA). A genetic algorithm relies on biologically inspired operators to find better solutions to an optimization problem. Each solution has properties that can mutate like chromosomes. A population of solutions is generally randomly generated at the beginning and random mutations are applied. Every time mutations are applied, a new generation of solutions is created. Then the best solutions in the generation are selected to constitute the basis for the next generation. The objective is to mutate and select solutions to improve the solution set at each generation. The article [18] used a GA to compute deployment plans but in this work, capacity limitations are not considered. [4] also used GA in a parallel system to improve computation performances. This approach is often appropriate for large problem instances where it is very difficult to improve a solution but the method is often slow to compute. In [4] the computation time is of several minutes.

In [22], an Evolutionary Algorithm Multi-Objective Evolutionary Algorithm (MOEA) and a Non-dominated Sorting Genetic Algorithm (NSGAII) are offered to solve a bi-objective Robust VRP.

The sub-problem optimization is studied in [34] that divides the problem into sub-problems by affecting demands to tours and then solving the problem in each tour. Hence the problem size is reduced and the computation time as well.

A Particle Swarm Optimization (PSO) algorithm is developed by [70] to solve a VRP with dynamic demands. The PSO starts from feasible solutions which are the particles. Each particle decides the direction to move in the search space at each iteration in regards to its best-known solution but also the best-known solution of the entire swarm.

Finally, [19] uses node-exchange and arc-exchange neighborhoods' local search to improve solutions.

Meta-heuristics reviewed in this section show good performances in terms of solution quality in a short amount of time compared to exact methods. However, it has been noticed, for Tabu search and GA for instance, that the computation time is still over a minute which is still too high for the crisis management context where decisions might need to be taken on the scale of seconds. For our problem, the use of heuristics seems more appropriate.

Online strategies

Online strategies are used to solve Dynamic and Stochastic VRPs. This type of approach is often offering lower solution qualities than other approaches but the computation is almost null. It is based on a set of rules called policies. They state how to route a vehicle according to its situation when it ends service at a node. For instance "The closest un-served customer" is a routing policy where the next customer is selected as the closest in the list of customers. It avoids building routes that would be modified by dynamic requests. For instance, in [78], this approach is used for a truck dispatching problem. In this problem, the requests and travel times are dynamic. Hence using routing strategies avoids the problem of pre-computed routes that would be affected by changing travel times. The next customer is computed on the fly depending on the current travel times and known customers.

The same approach is used in [3] on a Periodic VRP. This problem is stochastic since probabilistic data is available on uncertain information. The developed policies are compared to two basic policies in this paper.

This approach is very consistent with un-capacitated problems. Restocking does not have to be planned, and there are fewer reasons to prepare routes in advance as in [5].

The work in [97] also used online routing strategies to solve a Stochastic VRP with uncertainty on demands.

Re-Optimization approach

The Re-optimization approach is used to handle dynamic and Stochastic problems. The principle is to designate a heuristic that gives a solution to the static problem and compute it at different times of the time horizon to get an updated solution to the dynamic problem.

The term is used even when the solution used for re-optimization does not compute an optimal solution (heuristics for instance). The authors of [103] use the classical combination of an exact method coupled with re-optimization strategies for a PDP. A re-optimization strategy defines the conditions that lead to starting a re-optimization. It can be a temporal condition such as re-optimize at a fixed time interval for instance. But strategies may be defined on other criteria. Re-optimization may be started by:

- A given number of new demands.
- A total sum of quantity of new demands.
- The end of the service at a node.
- A new customer with certain characteristics.

The work [100] also uses a MILP formulation on a problem where travel times are time-dependent.

But the re-optimization approach may also be used to solve Stochastic problems. In [105], [89] and [9] solutions using Markov Decision Process coupled with re-optimization strategies are offered. A Markov Decision Process (MDP) is a formalization of some sequential decision-making problems under uncertainty. The definition of MDP has been inspired from [49]. At each time period $t \in \{1, \ldots, T-1\}$ during the finite time horizon, the decision-maker has to choose an action a_t from a set of actions A_t. The system is represented through states $s_t \in S$. Each action is associated with a reward r_t which depends on the state and the action chosen. An additional reward is granted according to the final state. This reward is noted r_T. A probabilistic transition function is defined. $p_t(s'|s,a)$ gives the probability of entering state s' knowing the current state s and the chosen action a. A policy $\pi = \{\pi_1, \ldots, \pi_{N-1}\}$ gives the action to choose at each time period. The accumulated expected reward at a time period t given a policy π and a state s is V_t^π so that:

$$V_t^\pi(s) = r_t(s, \pi_t(s)) + \sum_{s' \in S} p_t(s'|s, \pi_t(s)) \cdot V_{t+1}^\pi(s') \tag{2.11}$$

$$V_N^\pi(s) = r_T(s) \tag{2.12}$$

A MDP optimization problem consists in minimizing or maximizing the total reward $V_1(s)$ given a starting state s.

[52] presents a branch-and-regret heuristic that computes updated solutions by time intervals. At the beginning of each time interval, a solution is computed to complete routes for demands in the current interval.

In the following tables, a summary of the related work discussed in this section is given. Table 2.1 offers a summary by VRP characteristic and table 2.2 by solution approach.

The problem studied in this thesis regroups the following presented variations:

- Capacitated
- Time windows
- Heterogeneous fleet
- Split delivery
- Dynamic customers, requests, and travel times

The approaches developed in the thesis are:

- **Heuristics** for the static version of the problem
- **Re-optimization** and **online strategies** for the dynamic formulation

These choices are motivated by the computation time performances of the solutions presented in related works.

VRP Variations	Capacitated	Time Windows	Heterogeneous	Periodic	PDP	Split Delivery	Dynamic			Stochastic		
							Customer	Request	TT	Customer	Request	TT
Rodríguez-Martín et al. [82]	✓			✓								
Archetti et al. [7]	✓			✓								
Sungur et al. [95]	✓										✓	
Gendreau et al. [41]	✓									✓	✓	
Solomon [92]		✓										
Mitrović-Minić et al. [69]		✓			✓		✓					
Archetti et al. [8]	✓			✓								
Wen et al. [101]	✓			✓				✓				
Yang et al. [104]	✓										✓	
Hvattum et al. [51]	✓									✓	✓	
Bent and Van Hentenryck [15]	✓									✓		
Bent and Van Hentenryck [16]	✓	✓								✓		
Chen and Miller-Hooks [23]											✓	
Golden et al. [47]	✓		✓								✓	
Dror and Trudeau [31]	✓					✓					✓	
Gendreau et al. [43]							✓					
Wohlgemuth et al. [102]	✓				✓		✓		✓			
Gendreau et al. [44]					✓			✓				
Gendreau et al. [42]								✓				
Ichoua et al. [54]	✓									✓		
Larrain et al. [61]	✓			✓								
Larrain et al. [62]	✓	✓		✓			✓					
Berkoune et al. [18]	✓											
Allahviranloo et al. [4]		✓									✓	
Charris et al. [22]	✓										✓	✓
Erera et al. [34]	✓										✓	
Moghaddam et al. [70]	✓										✓	
Bianchessi and Righini [19]	✓				✓							
Afsar et al. [1]	✓											
Potvin et al. [78]	✓	✓						✓	✓			
Albareda-Sambola et al. [3]		✓								✓		
Angelelli et al. [5]				✓				✓				
Van Hentenryck et al. [97]											✓	
Yang et al. [103]	✓	✓			✓			✓				
Vu et al. [100]		✓							✓			
Zhu et al. [105]	✓										✓	
Secomandi and Margot [89]	✓										✓	
Archetti et al. [9]										✓		
Hvattum et al. [52]	✓									✓	✓	

Table 2.1: Related work summary by VRP variations (TT = Travel Times)

	Exact Method	Tabu Search Heuristic	Multiple-Horizon Heuristic	Genetic Algorithm	Multiple-Scenario Approach	VMND /L-S	PSO	MDP	Re-optimization Approach	Online Strategies
Rodríguez-Martín et al. [82]	✓									
Archetti et al. [7]	✓									
Sungur et al. [95]	✓									
Gendreau et al. [41]	✓									✓
Mitrović-Minić et al. [69]			✓							
Archetti et al. [8]			✓							
Wen et al. [101]			✓							
Yang et al. [104]					✓					
Hvattum et al. [51]					✓					
Bent and Van Hentenryck [15]					✓					
Bent and Van Hentenryck [16]					✓					
Chen and Miller-Hooks [23]					✓					
Gendreau et al. [43]		✓							✓	
Wohlgemuth et al. [102]	✓	✓								
Gendreau et al. [44]	✓	✓								
Gendreau et al. [42]		✓								
Ichoua et al. [54]		✓								
Larrain et al. [61]						✓				
Larrain et al. [62]						✓				
Berkoune et al. [18]				✓						
Allahviranloo et al. [4]				✓						
Charris et al. [22]				✓						
Moghaddam et al. [70]							✓			
Bianchessi and Righini [19]						✓				
Afsar et al. [1]						✓				
Potvin et al. [78]										✓
Albareda-Sambola et al. [3]										✓
Angelelli et al. [5]										✓
Van Hentenryck et al. [97]										✓
Yang et al. [103]	✓								✓	
Vu et al. [100]	✓								✓	
Zhu et al. [105]								✓	✓	
Secomandi and Margot [89]								✓	✓	
Archetti et al. [9]								✓	✓	
Hvattum et al. [52]	✓								✓	

Table 2.2: Related work summary by resolution approach (L-S = Local Search)

2.3 Simulation tools

In the literature, different articles have run simulations to validate their results on crisis management problems. This is the case in [68] on a large-scale earthquake.

Other works evaluate their results on cases study like [56]. The approach is different from simulation because here, the scenario of the case study is re-played identically. This evaluation approach is possible when several cases study are available.

Finally works like [92] evaluate their solution on benchmarks. In this example, the benchmark has been re-used by other works in the literature to validate their results and to compare the performances with the original solution.

In this thesis, since the studied problem has specific characteristics, benchmarks from the literature would not be used as such and would need to be modified. Therefore the comparison with other solutions from the literature would be biased. A case study is available for this thesis but we want to validate the solutions presented in several scenarios of various characteristics. It motivates the choice of the development of a graph generator that will enable the creation of instances of flooded territories and their issues with similar characteristics to the case study.

Finally, for the simulation of dynamic scenarios, tools from the literature are often not available and it would be difficult to integrate into the simulation process. The middleware from the Datazero project [75] answered the requirements of a simulation tool for this thesis and needed a few adjustments to be adapted into a simulator for DVRP. The details about this simulator are presented in chapter 3.

2.4 Conclusion

In this chapter, related works from the fields of crisis management and Vehicle Routing Problems have been presented. The crisis management literature offers various approaches to tackle problems from the response phase. Different articles studying the response phase of floods have been reviewed. However, none of the presented approaches fit the requirements for the victim relief operations after flash flood events. There are works on the evacuation of victims but not considering relief operations by rescue teams.

In the Vehicle Routing Problem literature, different problems with various characteristics have been reviewed. In table 2.1 the characteristics are summarized. From the presented works, none of them accumulates all the characteristics of the VRP studied in this thesis. The VRP for flash floods victims relief operations gathers the following variations:

- Capacitated
- Time windows

- Heterogeneous fleet
- Split delivery
- Dynamic customer, request and travel times

Different solution approaches have been presented in this chapter to solve the VRP. The following approaches are developed in this thesis and are discussed in the following chapters:

- Heuristics
- Re-optimization approach
- Online strategies

Note that for the rest of the manuscript, the following terms are used to be more adapted to a crisis management vocabulary:

- **Customers** is replaced by **Demands**. The term to describe customer locations are diverse:
  - Node
  - Demand point
  - Issue
- The word **request** is used only to describe the characteristics of a **Demand**.

Chapter 3

Experimentation tools and integration

The objective of this thesis is to develop solutions for the Dynamic Capacitated Vehicle Routing Problem under Deadlines (DCVRPD) and evaluate them. First, to model the territory for VRPs, a graph structure is often used in the literature. In this data structure, the vertices are the demand points where victims need to be rescued and the edges are road segments where vehicles are allowed to drive. To evaluate the different solutions tested in this work, graphs are generated to represent territories to experiment on. A graph generator able to create graphs with control characteristics is needed. This generator must be able to create graphs that represent a large spectrum of impacted territory. Furthermore, it needs to be coupled with a routine to select nodes inside a flooded area. This procedure is used with a shape of the flood and a graph of the impacted territory. Then it selects the nodes of the graphs that are inside the shape and returns the graph modified with nodes marked as demand points. The tool to cover that need has been developed for this work. There are not enough monitored flooding crises to test the solutions exhaustively. An evaluation of a few crisis scenarios would have little value since we could not confirm the resilience of the solution facing various types of problems. In addition, data is often incomplete after a flooding crisis – the emergency of the situation implies that logging the events for further study has a low priority – and scenarios of the flood cannot be replayed with fidelity. For this reason, a graph generator is necessary to model several types of territories. This tool is presented in section 3.1.

As stated above, the evaluation process in this work is made through several graphs and scenarios which own characteristics of real-life cases. A dynamic scenario is like a script that describes the evolution of the crisis over the first hours. A crisis scenario is composed of the different events that can happen during the crisis. In a dynamic environment, events happen during the crisis. For example, a vehicle might be delayed because the road it was supposed to drive through is blocked and it has to take another route. In such cases, events

have to be treated to adapt to the new situation. To assess the solutions that will be developed for this work on crisis scenarios, a simulator is needed. This tool should simulate the unfolding of the plan-making for vehicles while they evolve on their routes through time, and synchronize with the release of dynamic events in the scenario. It also needs to simulate the communication between the different rescue teams entities (decision center, vehicles, and call center) during a crisis. Finally, this software must play scenarios in simulated time to speed up the experimentation process. In section 2.3 softwares used in the literature are presented but none of them fit the requirements for simulation in this work. The software developed to play this role is presented in section 3.2.

3.1 Graph generator

The territory is represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the vertex set $\mathcal{V} = [\![0, V]\!]$ of size $V + 1, V \in \mathbb{N}$, where every vertex is a point of demand where people need to be rescued. This model is the most widespread in the literature (in [38] for instance). Let $\mathcal{E} = \{(i, j) : (i, j) \in \mathcal{V}^2, i \neq j\}$ be the set of the direct edges representing existing roads that link nodes together. As mentioned above, a generator is developed to generate graphs that model a territory. However, the objective here is not to reproduce a territory identically but to produce graphs similar to a type of modeled territory. Then the first question to answer is: how to create a parameterized model that could include most types of territories that could be impacted by a flood.

3.1.1 Territory model

A flood can impact various types of territories. Cities, as well as countryside areas, may suffer from floods. In both cases rescue operations are necessary. Hence, a model is needed that can represent a countryside territory as well as a big and highly populated city. Also, whatever the studied area, a territory is never uniform. For example, a city is often constituted of a center denser than its sub-urban areas. This is reflected in the model. From this observation, the choice has been made to model zones inside which density characteristics are constant. These zones are represented with concentric circles. This model is simplified compared to the actual shapes of territories. It could easily be modified to handle zones with different shapes. The model using circles seems satisfying for the evaluation process of this thesis and to produce graphs similar to the case study of Luchon.

Figure 3.1 shows how this model is applied to the city of Toulouse on a satellite image. In this image, one observes different zones corresponding to different population densities.

Along with population density, connectivity between two points also varies from one zone to another. In fact, in a big city center, several paths may be taken to go from point A to point B. Many streets might be used and this offers more options in case of a flood. If a path is flooded, if there are several route options,

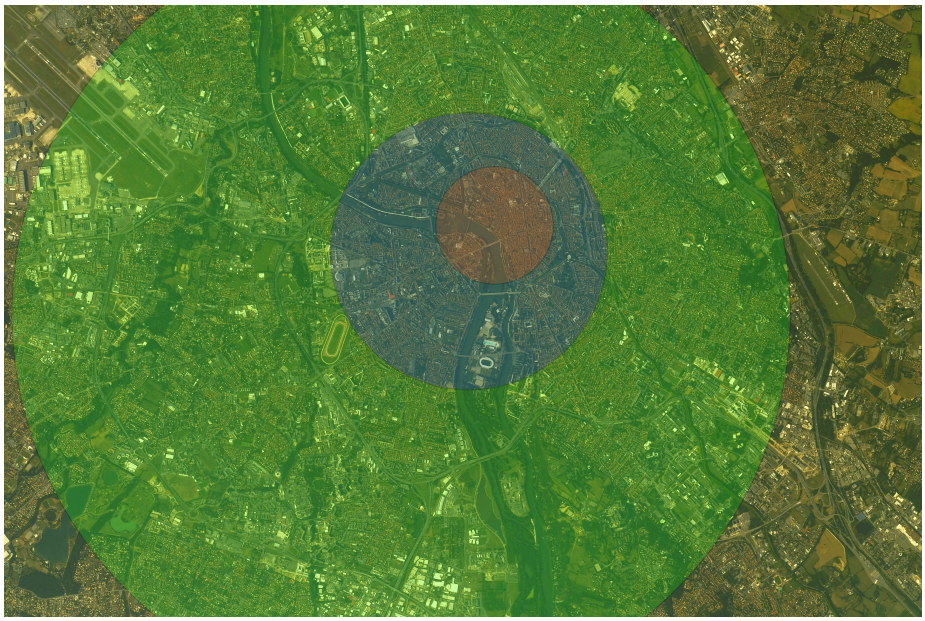


Figure 3.1: Satellite image of Toulouse from Copernicus program of ESA with density zones

another route may be still practicable. On the contrary, in a countryside area, there are more often situations where only one road can lead to a location. If this road is not practicable, different resources might be necessary.

The model of zones represented by circles is developed. In this model, if a circle overlays another it means that the bigger zone is represented by the bigger circle from which we remove the smaller circle. For 2 circles C_1 of radius r_1 and C_2 of radius r_2 with areas A_1 and A_2 respectively, so that C_2 overlays C_1, the zone areas A_1 and A_2 are defined as:

$$\begin{cases} A_1 = \pi \cdot r_1^2 \\ A_2 = \pi \cdot r_2^2 - A_{C_1 \cap C_2} \end{cases} \tag{3.1}$$

The model using these density zones answers to different types of territories. It can model an urban area as illustrated above, but also a countryside territory. If an area with no village is studied, one can model it as only one zone with low density. If the modeled territory is a big area of campaign with several small villages, it may be represented as several zones apart.

3.1.2 Territory topology

Coordinates of points of the territory are expressed in a Cartesian system with its origin at the rescue center. The unit of this system is the meter. As explained

above, the territory is represented through zones. These zones have different characteristics:

- **Center**: The center of a zone is a point in the Cartesian system. It is the coordinates of the circle's center that represents the zone area.
- **Size**: Radius in meters of the circle.
- **Density**: Density of population inside the area. However, instead of giving a population density, a density of potential issues per square kilometer is directly given to the generator. This value is considered to be proportional to the population density. The more an area is populated, the more it has housing infrastructures and public buildings.
- **Degree**: Represents the number of nodes from the graph that each node is directly connected to. It is used to build the edges of the graph. The degree of a node in the graph (number of edges that connect these nodes) equals the degree of its zone.

With these parameters, the graph generator works as described in the pseudo-code algorithm 1.

Algorithm 1
Pseudo-code for the graph generator process

```
function CREATE_GRAPH(zones)
    nodes ← [ ]
    edges ← [ ]
    for z ∈ zones do
        nbNodes ← compute_nb_nodes(z.size, z.center, z.density, zones)
        for i ∈ ⟦1, ..., nbNodes⟧ do
            nodes ← create_node(z.center, z.size, zones)
        end for
    end for
    for z ∈ zones do
        for i ∈ ⟦1, ..., nbNodes⟧ do
            nb_e ← z.degree − nb_edges(edges, i)
            edges ← create_edge(nb_e, nodes, edges, i)
        end for
    end for
    edges ← check_isolation(edges, nodes, zones)
    return nodes, edges
end function
```

Algorithm 1 uses:

- ***zones***: List of zones the graph is parameterized with. A zone is a tuple: (center, size, density, degree)

- ***compute_nb_nodes***: Computes the number of nodes to create in a zone. Parameters are the size and center of the zone. These parameters are coupled with the list of zones to compute the actual area of the zone. The list of zones is necessary to compute intersection with other zones. Then the area of the zone is combined with its density to compute the number of nodes to generate.

- ***create_node***: Creates a node and returns it. A node is a tuple of its coordinates in the Cartesian system. The center and size of the zone the node is created in are given as parameters. A list of zones is also required. All 3 are needed to compute the interval of coordinates values in the zone. In practice, the coordinates are first generated in a polar coordinate system, easier when working with circles. Then these coordinates are converted into the coordinates of the Cartesian system.

- ***nb_edges***: Computes the number of edges already existing for a given node. The parameters are the list of edges and the node **id**. This function searches in the list how many edges connect the node and returns this number. An edge in the list of edges is a tuple with the **id** of both nodes it connects.

- ***create_edge***: Creates a number of edges indicated by the first parameter. Edges are created with the closest nodes. The second parameter is the list of nodes. It allows computation of the closest nodes. The third parameter is the list of edges. It allows checking edges that already exist connecting the input node. In this case, the number of edges that need to be created for this node is reduced. The last parameter is the node **id**. Edges created are returned to add them to the list of edges. If one of the closest nodes already has enough edges, the next closest node is selected instead.

- ***check_isolation***: Checks if the creation of edges created isolated sub-graphs. This would mean that the nodes of these sub-graphs would not be reachable from the rescue center. This function checks if this problem exists using the lists of nodes and edges. If a sub-graph is found, it is solved by adding an edge from the closest node of the sub-graph to any node of the main graph.

In this algorithm, the creation of nodes and the creation of edges are in 2 separate loops since all nodes need to be created before starting to add edges. If we create an edge and then a closer node is added, the structure of the graph would not be as expected.

The graph is represented by a list of nodes and a list of edges. As mentioned above, a node is represented by its coordinates. Its **id** is its index in the list. Edges are also a list. When the graph is generated an edge is a tuple containing **ids** of the nodes it is connected to. Figure 3.2 shows an example of a graph that has been generated with the following parameters:

Zone	1	2	3
Center	0,0	0,0	0,0
Size (in meters)	1000	2000	4000
Density (nodes per km^2)	10	5	1
Degree	3	2	1

Table 3.1: Graph generation parameters

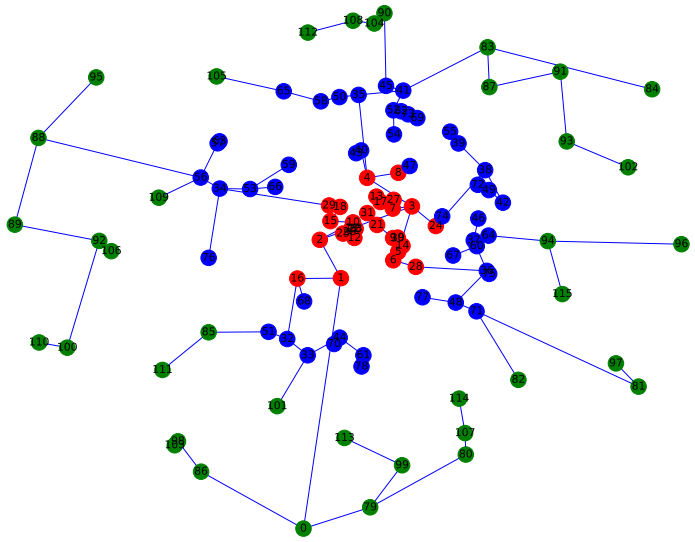


Figure 3.2: Graph generated example

In this graph example, the different zones are represented with different colors. The central zone is denser in nodes in this figure (nodes in red). On the opposite, the nodes from the external zone are sparser (in green).

The choice has been made to simplify the road segment as straight lines. In fact, in real-life, especially in cities, the path between two issues is not straight and turns have to be made, which changes the value of the distance associated with the edge. However, this simplifies the computation of distance associated with edges and is not problematic for evaluation since we generate territories and do not reproduce an existing one. For the final use, in i-Nondations project, these distances will be computed using the real road network. In this work, the simplification allows us to compute the distances associated with each edge using the coordinates of nodes only. Then the graph is transformed into a clique. A clique is a graph where any two nodes are connected with an edge. The valuation and the comparison of paths during graph traversal occur frequently in the algorithms that we develop, hence moving to a clique saves a lot of computation time. An exact method must avoid making the model too complex by visiting

nodes that are not served. This process is very usual in the literature. The route costs are computed using the shortest path between each node. Dijkstra's algorithm [29] is used to compute this shortest path. This algorithm computes the shortest paths in a graph and is a reference in the literature. From these shortest distances and with the travel speed of each category of vehicle, a route cost is computed for each edge of the graph's clique. These values are then stored in the route costs table.

3.1.3 Nodes selection

First, the graph of the territory is generated. Then a shape of the impacted area and a scale of the problem (number of nodes) are required. The generator applies the shape on the graph and randomly selects nodes inside the shape. The graph that the generator returns contain all nodes from the initial graph and a list of the index of nodes that are demand points. This section explains how the shape representing the flooded area is defined and how nodes are selected inside this area.

The shape defines the flooded area on the territory. The demand points of the problem are selected among the nodes from the graph that is inside the flooded area. In fact, the graph models the entire territory concerned with the flood, like a city and its surroundings for instance. But some potential issues may not be impacted by the flood due to their geographical situation for instance.

The impacted area is represented through a list of tuples. Each tuple is a portion of the segmented line that represents the flooded area. Coupled with the width of each segment, this segmented line creates a shape of the impacted area. Lets note x_r^i and y_r^i the coordinates of the point of index i of the segmented line, and w_r^i its width. A tuple at index i in the list contains the coordinates of a point x_r^i and y_r^i and the width of the portion in meters w_r^i. A line is drawn between the point of coordinates $[x_r^i, y_r^i]$ and the point of coordinates $[x_r^{i+1}, y_r^{i+1}]$. Then the width w_r^i is applied on both sides of this line to draw the shape of the impacted area between point i and $i+1$. The last element of the list has a width of 0 because no shape is to be drawn after this point. An example of shape is presented in fig. 3.3. The figure shows a simple graph with the impacted area. Nodes in green are selected to be points of demand. In this example, the scale of the problem is 4. Only 4 nodes are selected. The other nodes in the shape (in orange) and nodes outside the shape (in red) remain in the graph but are not characterized as demand points.

3.2 Simulator

During a crisis, events are released dynamically. Different types of events occur during the crisis and need to be included in the problem. Details about these events and their characteristics will be presented in chapter 5. According to these events, their nature, and characteristics, different decisions have to be

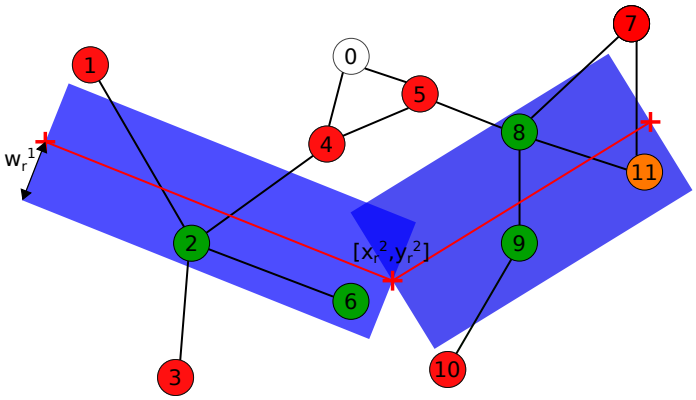


Figure 3.3: Example of a shape for nodes selection

taken. In this chapter, the discussion is on dynamic events in general. Events considered are released at a precise date in a continuous-time environment. Different simulators have been discussed in section 2.3, but none seems to fit the required characteristics listed below. Routes of interventions are prone to changes over time. To tackle this dynamic problem, reactive algorithms are developed in chapter 5. These algorithms need to be evaluated. A simulator has been developed for this purpose. This tool needs to fulfill several requirements:

- **Multi-Process**: The simulator should simulate the independent behavior of the different elements involved in flood relief operations. Several vehicles are in motion at the same time and they communicate with the Decision Center (DC) that gives them missions to execute. Since these actors are autonomous, each vehicle – as well as the decision center – is represented by a process. The last process that represents a call center that communicates dynamic events to the DC is also added to this multi-process environment.

- **Communication**: The different processes of the simulator need to communicate by messages. A bus of communication and a typology of messages need to be defined to that end.

- **Synchronisation**: Since several processes evolve on their own inside the simulator, a solution needs to be settled to make sure their operations are synchronized on the same clock. Furthermore, relief operations can last from several hours to several days. Then, a simulator that would run a scenario in real-time would lead to very high computation time for experiments. Furthermore, most of the time during relief operations, vehicles are evolving and no action is required. Hence dynamic scenarios of crisis may be run on simulated time.

The software developed to meet these specifications is presented in the sections below. The middleware that allows communication between processes and

their temporal synchronization (with a metronome) in a simulated time environment was originally built for the **Datazero** project [75]. This project works on a data center powered only by renewable energies. The software initially simulates the scheduling of tasks in the data center to evaluate strategies to reduce energy consumption. It has been adapted to fit the simulation needs of DVRP.

3.2.1 Multi-process

Different types of processes are defined to simulate the rescue teams entities of the relief operations. Each type of process has its own behavior and own code. The following types of process are defined:

- **Decision center**
- **Vehicle**
- **Event generator**

Decision center

This process gathers information about the different events to compute the routes dynamically. It receives different events from the event generator and vehicle processes and computes updated routes. These new routes are sent to vehicles in the form of missions to be served. This process's routine is presented in algorithm 2.

Algorithm 2
Decision center process algorithm

Input : Size of the problem: n
$events \leftarrow [\,]$
$logfile \leftarrow [\,]$
$graph \leftarrow generate_graph(n)$
$create_processes()$
while $!victim_all_rescued()$ **do**
    $events \leftarrow wait_for_event()$
    $logfile \leftarrow log(events)$
    $graph \leftarrow update_graph(events)$
    $routes \leftarrow update_routes(events)$
    $send_missions(routes)$
end while
$terminate_processes()$
return logfile

The call to $generate_graph(n)$ generates a graph as described earlier in this chapter, passing the scale of the problem (number of nodes) as an argument.

More details about the operations in *update_graph*(*events*) and *update_routes*(*events*) are given in chapter 5.

Vehicle

This process simulates the behavior of a rescue vehicle during a crisis. Created by the *decision center* process, a vehicle receives missions, performs them, and waits for the next missions. Missions are objects that contain different fields. The different fields of a mission are:

- **arrivalDate**: the date when the vehicle is planned to arrive at the node (noted as mission.arrivalDate).

- **endService**: the date when the vehicle is expected to terminate fulfilling a service at the node (noted as mission.endService).

Other fields are included in mission objects and are specified below in section 3.2.2. When there is no mission to be served, it waits for missions at the depot. It is then considered in *pending* status. The behavior of any vehicle process is described in algorithm 3.

Algorithm 3
Pseudo-code used for vehicles processes

```
mission, vehicle ← receive_mission()
while mission ≠ none do
    if mission ≠ none  &  vehicle ≠ pending then
        sleep_until(mission.arrivalDate)
        send_arrival_event()
        sleep_until(mission.endService)
        send_end_service_event()
    end if
    mission, vehicle ← receive_mission()
end while
```

Algorithm 3 uses:

- **sleep_until**: This function in the pseudo-code symbolizes the role played by the metronome that synchronizes the processes. Its behavior is described in section 3.2.3.

- **send_arrival_event** and **send_end_service_event**: These functions are used to send an event to the DC. These events signal arrival to the node and the end of a service at a node respectively. The way messages are exchanged and their typology is described below.

Event Generator

This process represents all entities in real life that communicate data to the DC. It can be scouting vehicles or victims that call for rescue for instance. This process generates the dynamic events and sends them to the DC. Dynamic events are released over the time horizon of the crisis. The time period that defines the crisis is called the time horizon and is noted T. Events are not described exhaustively and in detail in this chapter and will be specified in chapter 5. However, their common characteristic is their release date (the date the event is known). This information about event object e is accessed through $e.release_date$. The code of this process is simplified in algorithm 4.

Algorithm 4
Event generator process algorithm

Input: time horizon: T
$events \leftarrow generate_events(T)$
while $events \neq [\,]$ **do**
  $e \leftarrow events[0]$
  $sleep_until(e.release_date)$
  $send(e)$
  $events \leftarrow delete(events, e)$
end while

In this section, communication between processes has been abstracted for the sake of simplicity. The following section presents the underlying architecture that drives communication between processes.

3.2.2 Communication

The communication between processes is operated through 2 components: ActiveMQ which plays the role of the communication bus and Protobuf which is used to format the exchanged messages on this bus.

ActiveMQ bus

Apache ActiveMQ [6] is an open-source message broker. It is based on Java and coupled in the software with `stomp.py` as a protocol. Once ActiveMQ is configured, it plays the role of relay between processes. Every process has a message *sender* and a message Listener. These *sender* and *listener* subscribe to *topics*. A topic is a type of message with a certain format. The format of the messages is defined using Protobuf, presented in section 3.2.2 below. When a process subscribes to a topic, its listener receives the messages of the *topic* that are sent on the bus. Then the process deals with it according to its characteristics. Figure 3.4 shows the architecture of communication using the ActiveMQ bus.

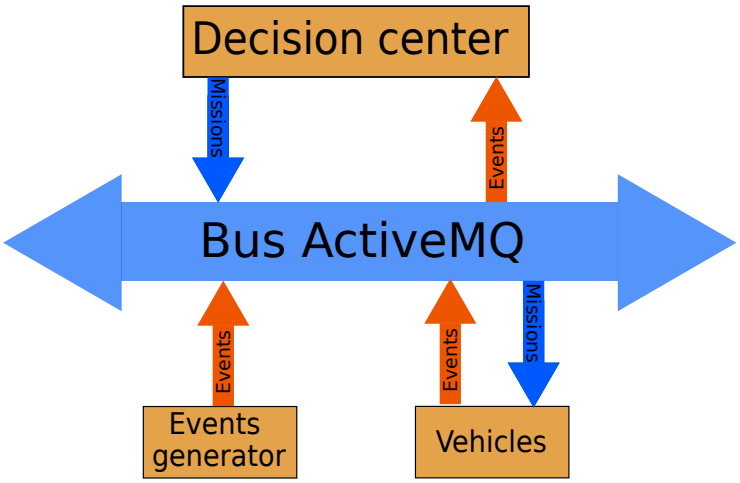


Figure 3.4: ActiveMQ bus architecture

Protobuf messages

Protobuf for Protocol Buffer [28] is a serialization format of messages developed by Google Developers. Serializing messages allows communication with generic messages through a socket. To illustrate how Protobuf messages are defined, *mission* topic is used. Various types of events are also defined and detailed in chapter 5. These topics are:

- Mission
- Update
- New demand
- Delay
- Resource modification

The field of a message is a primitive type. Complex types can be defined by the repetition of a primitive type or a defined message.

A mission is defined as follow:

- int32: **vehicle**: The vehicle that executes the mission.
- int32: **nodeId**: Id of the node to serve
- int32: **quantities**: The number of victims to rescue at the node during this mission

- fixed64: **arrivalDate**: The planned arrival date of the vehicle at the node according to the routes
- fixed64: **serviceTime**: Estimated time to rescue victims at the node
- int32: **previousNode**: Last visited node before this mission
- int32: **nextNode**: Next node to visit after this mission
- int32: **turn**: The tour of the mission in the routes. It allows identification of the mission in the routes if the node is served twice by the vehicle in different turns.
- fixed64: **timeStamp**: instant when the message has been sent. This field is common in all types of messages.

The timestamp is an important field that allows synchronizing messages reception with their sending in a multi-process environment. The synchronization process is operated by the metronome. It is presented in the section below.

3.2.3 Synchronization

In a parallel system such as this simulator, processes need to be synchronized. To do so, a metronome is integrated into the simulator. This metronome allows processes to give a date it wants to synchronize to and waits to be wakened up. The metronome records this date and keeps it in a table with dates from all processes. Once every process gave its date, the metronome selects the process with the closest date and wakes him up. This process executes its actions and synchronizes on its next date.

This routine runs while all processes are not terminated. This also allows for running dynamic scenarios in simulated time. When the metronome chooses the next date, a jump forward in time is operated on the common clock. Thanks to that, a scenario can be run quicker than the actual duration of the crisis. Since a lot of scenarios will be simulated in this work it is necessary to gain time on experiments.

Figure 3.5 offers a sequence diagram that displays interactions between processes. This diagram is an example with only one represented vehicle. The sequence begins with the update of new routes. In this figure, it can be observed that there is concurrent access to the current routes. Vehicles execute missions while DC tries to compute new routes. The missions are computed from the routes. Concurrent access in this context might mean that the vehicle serves a mission that has been re-computed and therefore does not exist anymore for the DC. To avoid such issues, locks are put on the variables that describe the current routes. Another mechanism allows for avoiding questioning missions that are being served. It is defined in chapter 5.

In this figure, the example starts with a routes update. This update might have been triggered by any type of event. Routes are updated and the DC listener is signaled that new missions are available. Then it sends a mission from

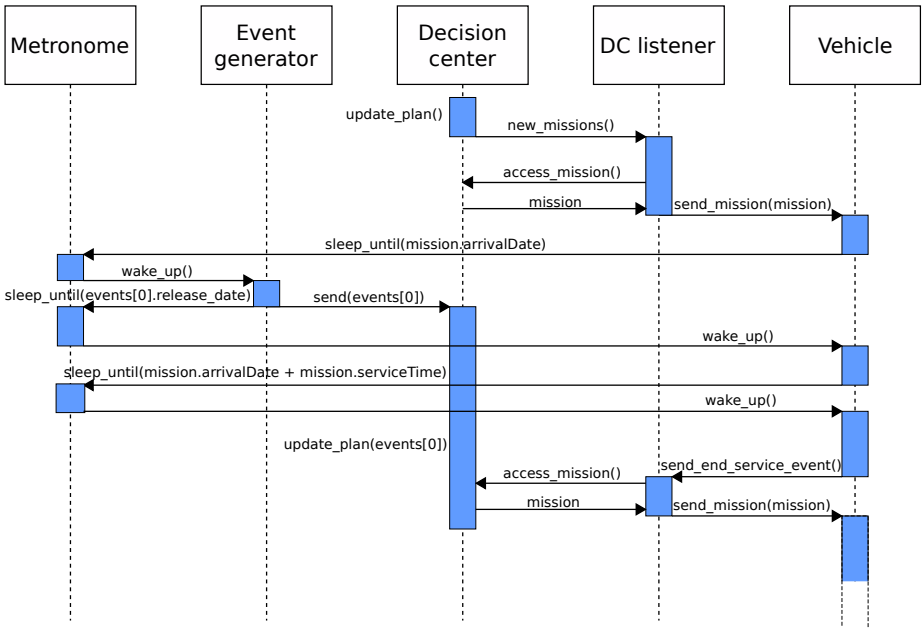


Figure 3.5: Simplified sequence diagram of the simulator

the new routes to the vehicle. The vehicle receives it and gives its synchronization date to the metronome and goes asleep. Then the metronome picks the next date and wakes up the event generator. This process sends its event, selects the next event, and synchronizes with its date of release. The metronome then synchronizes on the next date which is the arrival of the vehicle at the node. The vehicle wakes up and synchronizes on the date when the service ends. It then wakes back up at the date of the end of service. The vehicle signals the end of service to the DC which selects the next mission for the vehicle and sends it. This process is repeated until all events are flushed and all missions are served.

3.3 Conclusion

In this chapter, different tools that are used in this thesis are presented. These tools are developed to help the evaluation process of the solutions developed for the problem. These softwares have been developed because the different tools from the literature do not satisfy all the requirements.

The first developed software is a graph generator. It creates graphs that model flooded territories of various characteristics. Then it enables the selection of nodes inside a flood shape. These nodes are turned into demands.

The second software is the simulator. It is used to simulate dynamic crisis scenarios. It models communications by a message between processes that represent the different actors of the crisis. It also helps assess solutions to scenarios

in simulated time to speed up the evaluation process.

The tools presented in this chapter have been introduced with general notions. More details about some aspects such as dynamic events or characteristics of the studied territories will be detailed in further chapters.

Chapter 4

Static problem

In chapter 1, it has been developed that the problem studied in this thesis is a VRP. In this chapter, the studied problem is the response phase of the crisis management of a flood. Victims' relief operations during a flood are studied. The type of flood this work focuses on is flash floods. A flash flood is characterized by a very fast increase in the water level. Hence, there is very little time for preparation. The studied problem is a static VRP. This problem relies on the hypothesis that all information about the problem is known in advance. It matches the limit between the *Preparedness* and *Response* phases of the relief operations in the disaster management literature. Routes are defined for the relief operations and these routes are executed without modifications.

In a VRP, the objective is to optimize the routes of rescue vehicles to serve demand points. At these demand points, rescue teams save victims and take them in the rescue vehicles. However, vehicles have limited capacities. The number of victims that can be relieved by a vehicle has to be considered. VRPs where capacity limitations of vehicles are considered are called Capacitated VRP. When a vehicle has reached its capacity limit, it has to go back to the rescue center called depot to put the victim to safety. This operation is called restocking. After restocking, the vehicle can start a new route with its full capacity. The part of routes of a vehicle between two drives through depot is called a tour. Due to the disaster management context, rescue teams operate in a degraded state. A lot of victims have to be relieved and rescue teams are understaffed. This means they do not possess enough resources to solve the problem in one tour. In this problem, the depot is both the place where victims are taken once rescued and the rescue center where operations are decided.

In this VRP, victims' relief operations are studied. The relief of a victim may be more or less urgent according to the situation. A deadline is associated with demands. It is the date when victims must be rescued the latest. After this date, the health integrity of victims is not guaranteed. Therefore, the studied problem is also a Vehicle Routing Problem with Time Windows (VRPTW). A priority is also associated with each demand to help rescue teams order their interventions. The priority coefficients and deadlines values have been defined by rescue teams

from SDIS 31 (firefighter entity in charge of the relief operations during floods). They have been presented in chapter 1. In the conducted experiment, priority and deadlines values are associated but in real-life situations, these variables can evolve separately.

Rescue teams need to save victims using a fleet of vehicles. These vehicles are the resources of the problem. During a flood, various types of vehicles are used. In this work, 5 categories of vehicles are defined and presented in chapter 1.

Each category of resource is associated with a mean travel speed which determines the travel times for each category. This category of vehicles defines the various type of resources to access the different areas where victims need to be rescued during a flood. According to the situation of locations where victims need to be rescued, a specific type of resources is necessary. The type of VRP that studies problems with several categories of vehicles is Heterogeneous Fleet VRP.

During a flood crisis, several areas are often impacted. For instance, different cities on the same river can be flooded at the same time. Rescue teams need to dispatch their resources among the different impacted sectors. Resources may also need to be dispatched to a single sector. For example, the flood can separate the impacted area into two or more sub-areas. In this case, a depot has to be created on both sides of the water and resources must be dispatched to each depot. This problem can be generalized to more than two depots. The different sub-problems created in these cases are called clusters. The distribution of resources (firefighters and vehicles) between clusters can be optimized in order to give the best response to both sectors.

The hypotheses of the static problem are:

- Travel times are inputs of the problem and are static. They do not evolve during the crisis.
- Demands can be split. A given demand can be served by several vehicles.
- All points of demand are connected. There is no demand point that cannot be accessed by rescue teams.
- A vehicle route must start and end at the depot. Rescue teams' operation base is the depot, hence vehicles start from there and need to get back to drop rescued victims to safety at the end of their tour.

To solve this problem, different approaches are studied in the literature. Exact methods can be used but are not adapted to every context. If the problem is too complex and the solution needs to be computed in a short time, an exact method might not be appropriate because it would take too long to find a solution. For example, in [10], solutions for a VRPTW and a CVRP are computed on the scale of hours and not a few seconds for test instances of 20 nodes. A Mixed Integer Linear Programming formulation is presented. This method is tested on different problem sizes to evaluate if it answers the computation time requirements implied by the crisis management context. The complexity of the problem is also studied. It is known from the literature that NP-complete

problems are difficult to solve with exact methods in real-time, especially when they are NP-complete in the strong sense. For this reason, heuristics have been developed. They aim at finding the best possible solution to the problem while keeping a short computation time. These solutions are evaluated to determine which fits the most the problem requirements, in terms of solution quality, computation time, and reliability.

In this chapter, the static problem is studied. In section 4.1, the problem is described in details. The mathematical model is presented. In section 4.2, developed solutions to answer the static VRP are presented. Then the clustering problem, faced by rescue teams when dealing with several sectors in parallel, is presented in section 4.3 and solutions are developed. Finally section 4.4, presents the experiments conducted. The generation processes for experimentation are detailed and the experimental results are analyzed.

4.1 Problem description

In this chapter, we study the Capacitated Vehicle Routing Problem under Deadlines (CVRPD) for flash floods relief operations. The objective is to optimize routing of rescue vehicles. Rescue vehicles must save victims at demand points. The problem is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the vertex set $\mathcal{V} = [\![0, V]\!]$ of size $V + 1, V \in \mathbb{N}$. Every vertex is a point of demand. The depot (rescue center) for this VRP, is the vertex 0 of the graph. $\mathcal{V}^\star$ is defined as the set of all vertices without the depot. In this graph, the roads that link demand points are represented through $\mathcal{E} = \{(i, j) : (i, j) \in \mathcal{V}^2, i \neq j\}$ the set of the direct edges.

The following inputs are defined to describe the problem:

- Each vehicle has a category. The category is noted:
$$c \in \mathcal{C} = [\![1, 5]\!] \tag{4.1}$$
- The graph is weighted: a cost is associated with each edge. This cost is the route cost (travel time). The graph is a multi-layer graph, each layer matches with a category because route costs depend on the category of vehicle used. Furthermore, some road segments are not practicable for some categories of vehicles so the route cost for these edges is infinite in the corresponding layer. The travel time is noted:
$$tt_{i,j,c} \quad \forall (i, j) \in \mathcal{V}^2, c \in \mathcal{C} \tag{4.2}$$
- Every demand has a size (quantity or demand), the number of victims to rescue at a node i:
$$d_i, \quad d_i \in \mathbb{N} \quad \forall i \in \mathcal{V}^\star \tag{4.3}$$
- When rescue teams serve a node, a time is dedicated to the operation. This time is called service time. It may also represent the time necessary to drop victims at the depot. The service time for a node i is noted:
$$a_i, \quad \forall i \in \mathcal{V} \tag{4.4}$$

- The date at which information is revealed about demand is the release date. In the VRPTW literature, this date is the beginning of the time window and information may be revealed before the release date but the demand cannot be served before the release date. In this problem, once the information (victims need to be rescued) is known, relief operations can start. Therefore the release dates are reveal dates. They are noted:

$$r_i, \quad \forall i \in \mathcal{V}^\star \tag{4.5}$$

  Note that for the static problem, since all information is known at the beginning of the problem, release dates are null.

- Demands are served with vehicles. The fleet of available vehicles is composed of vehicles of different categories. The category of a vehicle k is:

$$cat_k, \quad \forall k \in \mathcal{M} \tag{4.6}$$

  With $\mathcal{M}$ the set of vehicles.

- As mentioned above, each demand is also associated with a category.

$$c_i, \quad \forall i \in \mathcal{V}^\star \tag{4.7}$$

  A node of a given category can only be served by a vehicle of the same category. The category of each node is an input of the problem. It is determined by rescue teams according to information about demands.

- Vehicles have a limited capacity which may lead them to restock. The term restocking is used in the literature to describe a vehicle that drives back to the depot to empty its load or to refill its reserve before starting a new tour. The capacity of a vehicle k is:

$$Q_k, \quad \forall k \in \mathcal{M} \tag{4.8}$$

- The crisis management context implies that the studied VRP is under a degraded state: there are not enough available resources to serve all demands in a single tour. Vehicles probably have to serve demands, drive back to the depot to drop victims to safety, and start another tour. Tours are indexed by $z \in \mathcal{Z}$.

- The priority of a node i is:

$$p_i, \quad \forall i \in \mathcal{V}^\star \tag{4.9}$$

  This is an input of the problem, estimated by rescue teams according to their knowledge about the demand. The priority relies on several factors:

  – Is the demand for a building that hosts public, for instance, hospitals, schools, or retirement homes? These types of building host fragile people that need to be evacuated quickly to avoid complications.

- The time of the day also plays a role in prioritization. If the crisis happens during the night, a school is not an issue to worry about whereas, during the day on weekdays, these locations are filled with potential victims.
- The state of health of the victims is considered. Most of the time, it can only be estimated with information by distance. If a victim is seriously injured though, its priority is higher than thus of a healthy person.
- The age of victims is taken into account. Children and seniors are considered to be more vulnerable. An old person cannot sustain the discomfort of water in their house as younger adults can.
- The architecture of the location is a major criterion. If the demand point is a house with several floors, victims can find refuge in the higher levels. On the opposite, if it is a ground floor house, the priority of the demand is higher.
- Water height at the demand is also a major factor. If the water is very low and does not get into the household that requires intervention, the discomfort for victims is reduced and they may wait longer before being rescued. On the opposite, if the water is 80 centimeters high in a house and victims cannot take refuge on higher grounds, the priority of the demand is higher.
- The season of the flood is an important parameter to consider. Being flooded with 40 centimeters of water during the summer seasons will not require fast intervention whereas the same situation during winter is harder to sustain long for victims.

These factors are not exhaustive, though it gives an idea on how priorities are affected to demands. To each of the 4 priority values, a deadline is associated. Deadline values are computed according to the priority and the release date and noted:

$$f_i, \quad \forall i \in \mathcal{V}^\star \tag{4.10}$$

The values for deadlines according to the priority are:

1. 12 hours: Lower priority category where the victim could remain on the spot. However, since the time horizon of a crisis is often lower than 12 hours, this value has been set to avoid making victims wait too long and avoid them too much discomfort.
2. 12 hours: Corresponds to the priority factor of SDIS 31.
3. 6 hours: Value of the priority coefficient as well.
4. 3 hours: This high priority factor is not associated with a deadline value in rescue teams' processes. It corresponds to "Must be rescued in emergency". This value has been set to avoid infeasible problems by setting a value too low.

This value is added to the release date value to obtain the date of the deadline.

Finally, variables need to be defined to describe routes. Routes are composed of tours for all vehicles. For each vehicle, on each tour, they describe:

- Which nodes it serves and in which order.
- The expected date of arrival at nodes.
- The number of victims to rescue at the node.

Before setting the variables, it must be specified that to simplify the model, the graph is transformed into a clique. A clique is a type of graph where every node is connected to any other node. To transform a graph into a clique, an edge is built between every two nodes that are not already connected initially. To do so the route cost is calculated by computing the shortest path. Once the shortest path is found, it is translated into a travel time and recorded in a $tt_{i,j,c}$. Three decision variables are set:

- $x^z_{i,j,k}$ is a binary variable equal to 1 if vehicle k travels from node i to node j on tour z.
- $h^z_{i,k}$ is an integer variable equal to the expected date of arrival of vehicle k at the node i on tour z. This variable is null if the node i is not visited by vehicle k on tour z.
- $q^z_{i,k}$ is an integer variable equal to the number of victims that are rescued by vehicle k at node i on tour z.

The crisis is finite in time. The duration of the operations is called the time horizon and is denoted T.

4.1.1 Mixed Integer Linear Programming

The objective of this problem has been determined through discussion with the rescue teams from SDIS 31. Rescue teams aim at reducing the time victims wait before being rescued. This time is called Flowtime. Furthermore, the higher the demand priority, the earlier one wants to serve it. Hence, the flowtime is weighted by priority. In addition, victims are considered individuals. Even if they are regrouped on points of demand, the flowtime is individual and it is not equal to the latest flowtime for the node. However, to keep the model linear and avoid quadratic terms for resolution, we do not want to add the quantity to the objective. Therefore, for the moment the objective is simplified and does not contain the quantity. It will be individualized later in the manuscript when the exact method is not considered. This simplification discourages split demands:

$$\min \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{M}} \sum_{z \in \mathcal{Z}} (h^z_{i,k} - r_i) \cdot p_i \tag{4.11}$$

subject to:

$$\forall i \in \mathcal{V}, \quad \sum_{k \in \mathcal{M}} \sum_{z \in \mathcal{Z}} q_{i,k}^{z} = d_i \tag{4.12}$$

$$\forall i \in \mathcal{V}^{\star}; k \in \mathcal{M}; z \in \mathcal{Z}, \quad h_{i,k}^{z} - r_i - R \cdot (1 - \sum_{j \in \mathcal{V}} x_{j,i,k}^{z}) \leq f_i \tag{4.13}$$

$$\forall j \in \mathcal{V}; k \in \mathcal{M}; z \in \mathcal{Z}, \quad \sum_{i \in \mathcal{V}} x_{i,j,k}^{z} - \sum_{i \in \mathcal{V}} x_{j,i,k}^{z} = 0 \tag{4.14}$$

$$\forall k \in \mathcal{M}; z \in \mathcal{Z}, \quad \sum_{i \in \mathcal{V}} q_{i,k}^{z} \leq Q_k \tag{4.15}$$

$$\forall i \in \mathcal{V}; j \in \mathcal{V}; k \in \mathcal{M}; z \in \mathcal{Z},$$

$$h_{i,k}^{\phi(i,z)} + a_i + tt_{i,j,cat_k} - R \cdot (1 - x_{i,j,k}^{z}) \leq h_{j,k}^{z} \tag{4.16}$$

$$\forall j \in \mathcal{V}^{\star}; k \in \mathcal{M}; z \in \mathcal{Z}, \quad \frac{q_{j,k}^{z}}{Q_k} \leq \sum_{i \in \mathcal{V}} x_{i,j,k}^{z} \tag{4.17}$$

$$\forall k \in \mathcal{M}; z \in \mathcal{Z}, \quad \sum_{j \in \mathcal{V}^{\star}} x_{0,j,k}^{z} \leq 1 \tag{4.18}$$

$$\forall z \in \mathcal{Z}^{\star}; k \in \mathcal{M}, \quad \sum_{i \in \mathcal{V}^{\star}} x_{0,i,k}^{z-1} \geq \sum_{i \in \mathcal{V}^{\star}} x_{0,i,k}^{z} \tag{4.19}$$

$$\forall j \in \mathcal{V}^{\star}; k \in \mathcal{M}; z \in \mathcal{Z}, \quad \sum_{i \in \mathcal{V}} x_{i,j,k}^{z} \times c_j = cat_k \tag{4.20}$$

Where

- $\phi(i,z) = \begin{cases} z-1 & \text{if} \quad i = 0 \\ z & \text{otherwise} \end{cases}$
- R is an integer of big size compared to all other variables. We will use $R = 10^{12}$.

A reminder of the inputs and variables of the model is presented below:
The objective function and the role of each constraint are detailed below:

- The objective function is stated in eq. (4.11). It is the sum of the flowtimes for all interventions, weighted by the number of victims rescued and the priority of the demand.

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	Service 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)
T	Time horizon of the crisis

Table 4.1: Inputs

$x^z_{i,j,k}$	Binary variable equal to 1 if and only 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
$q^z_{i,k}$	Number of victims taken by vehicle k at node i on tour z

Table 4.2: Variables

- Equation (4.12) covers the service constraints. It ensures that all demands are served entirely. It states that the sum of the rescued victims is equal to the demand, for each node.
- Deadline constraints are described in eq. (4.13). These constraints state that the service of a demand must be done before its deadline. These constraints are applied to all nodes, all tours, and all vehicles. If the node i is served by vehicle k on tour z, the term where R is included is equal to 0 and the constraint forces the flowtime not to be greater than the deadline. Else if the demand is not served, the constraint is ignored because the flowtime is negligible compared to the term with R and the constraint is always true.
- Constraints in eq. (4.14) make sure that a vehicle that arrives at a node also leaves it.
- The capacity limitation constraints in eq. (4.15) state that for every vehicle and each turn, the sum of the rescued victim is lower than the vehicle capacity.
- Constraints in eq. (4.16) are the planning consistency constraints. It states

that all consecutive missions in the routes respect the timeline. The arrival date to the second node is greater than the arrival at the first node plus the service time.

- Equation (4.17) states that if victims are relieved at a node (indicated by variable q), the vehicle must go through the node (indicated by variable x).
- Constraint eq. (4.18) ensures that every vehicle leaves the depot only once a tour. It avoids planning several tours for a single tour index.
- The planning continuity is described by eq. (4.19). It makes sure that the planning does not have empty tours.
- A demand of a given category can only be served by a vehicle of the same category. It is stated in eq. (4.20).

4.1.2 Complexity

We state that 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 [40]. A 3-PARTITION problem consists in deciding whether a set $\Gamma = \{b_1, ..., b_N\}$ of $N = 3n$ positive integers can be partitioned into n triplets $\Gamma_1, ..., \Gamma_n$ (where for any $k \in \{1, \dots, n\}$, $\Gamma_k = \{g_{k,1}, g_{k,2}, g_{k,3}\}$) such that $\sum_{i=1}^{3} g_{k,i} = B$. We will denote $\sigma : \{1, \dots, n\} \times \{1, \dots, 3\} \to \{1, \dots, N\}$ the permutation such that for all $(k, i) \in \{1, \dots, n\} \times \{1, \dots, 3\}, g_{k,i} = b_{\sigma(k,i)}$.

First, CVRPD is NP since one can check in a polynomial time whether given routes is feasible or not. From any 3-PARTITION problem instance that 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 $\mathcal{V}^\star = \{1, \dots, N\}$ of N demands, whose service 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 are from the same category numbered 1 and the priority for all demands equals 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: $\forall i \in \mathcal{V}^\star, r_i = 0$. We ask whether there exists a solution with a flowtime that is less than $N \times 9B$

($\Rightarrow$) 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 $k \in \{1, ..., n\}$, $\sum_{i=1}^{3} g_{k,i} = B$, and we build a solution to I_2. For every vehicle $k \in \{1, \dots, n\}$ 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 $j \in \{1, \dots, N\}$, hence the node $\sigma(k, 1)$ (resp. $\sigma(k, 2)$, $\sigma(k, 3)$) needs an service 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 $k \in \{1,\dots,n\}$. 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 $k \in \{1,...,n\}$, $x^z_{i,j,k} = 0$ otherwise. The variable $h^z_{i,k}$ as to be affected according to the order of the routes 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 plans exactly 3 interventions. We define a permutation σ such that for all $k \in \{1,\dots,n\}$ as an intervention on nodes $\sigma(k,j)$ for $j \in \{1,\dots,3\}$. With the deadlines $\forall i \in \mathcal{V}^\star f_i = 9 \cdot B$, by removing the travel times we have $\forall k \in \{1,\dots,n\}, \sum_{j=1}^{3} a_{\sigma(k,j)} \leq B$. 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 $\forall k \in \{1,\dots,n\}, \sum_{i=1}^{3} b_{\sigma(k,i)} = B$. Therefore, I_2 has a solution if and only if I_1 has a solution.

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

4.2 Heuristics

Due to the crisis management context, solutions to the problem must be computed within a limited time. Rescue teams cannot afford to wait hours before getting a feasible solution. The problem has just been proved to be NP-complete. Finding the optimal solution with an exact method can take a lot of computation time. The use of a MILP solver is assessed in section 4.4. In the meantime, when one wants to compute solutions to a VRP in a short amount of time, heuristics can be used. This category of algorithms does not guarantee finding the optimal solution. On the contrary, the objective is to find a feasible solution, the best possible, in the shortest computation time. For the rest of the manuscript, in cases when a heuristic does not achieve to find a feasible solution to an instance of the problem, we will call this event an **error**. This term will also be used for the exact method if a feasible solutions is not found **in the allocated computation time**. Different types of heuristics are presented in this section and will be evaluated later.

4.2.1 Greedy algorithms

The first type of proposed heuristics is greedy algorithms. A greedy algorithm creates routes step by step under locally optimal choices according to a defined criterion. At each step, however, past decisions are not questioned while inserting new demands in the route of a vehicle. These algorithms have been adapted from first fit algorithms used to solve the bin packing problem [48].

Four greedy heuristics are presented, each one uses a different criterion:

- **Shortest Distance Insertion (SDI)**: Demands are inserted by order of distance. Once a demand is inserted, the next demand to insert is the closest demand to any available vehicle. This criterion and heuristic are considered the most appropriate way to model rescue teams' behavior.
- **GreedySize Increase**: Demands are inserted by increasing size order. At each step, the smallest remaining demand is selected to be inserted next.
- **GreedySize Decrease**: Demands are inserted by decreasing size order. The biggest left demand is selected to be served next.
- **GreedyAlea**: Demands are inserted randomly into the routes. This heuristic has been developed as a baseline to compare the other greedy heuristics.

This criterion is used to choose between demands of the same priority, but demands are still inserted in the priority order first. Once a demand has been selected, it is served entirely according to vehicles' capacity limitations. Only then the next demand is selected. While the criteria of the 4 greedy algorithms are different, the rest of the algorithm follows the same structure. The greedy algorithms structure is presented in algorithm 5.

Algorithm 5
Greedy algorithm behavior routine

```
Input: d: list of demands
l ← sortDemands(criterion, d)
while l ≠ [ ] do
    while l[0] > 0 do
        vehicle ← firstAvailableVehicle(l[0], criterion)
        insertDemand(vehicle, l[0])
    end while
    l ← sortDemands(criterion, d)
end while
```

Algorithm 5 uses:

- **sortDemands**: This function sorts demands according to a given criterion. It returns a sorted list of demands.

- **firstAvailableVehicle**: Returns the first available vehicle in $\mathcal{M}$ for a given demand. First, the vehicle is selected according to the criterion. For example, if the criterion is the distance, the selected vehicle is the closest to the demand. In the other cases, the vehicle is the first available in the planning. Then, the function verifies if the insertion of the demand in the route of this vehicle would violate a deadline. If it would, the next available vehicle is selected and the checking routine is applied to it, and so on until a valid vehicle is found. If no vehicle is found, the algorithm returns an error indicating that the heuristic did not find a feasible solution.

- **insertDemand**: Inserts a demand in the vehicle. The demand is inserted at the end of the route. If the demand to insert exceeds the left capacity inside the vehicle for the currently planned route, only the quantity to fill the vehicle is inserted. Otherwise, the demand is inserted entirely. If the vehicle is full after insertion, the route is updated to drive the vehicle back to the depot and empty it. Then, after insertion, the vehicle needs to be re-selected if the demand is not served entirely. Since the vehicle is back at the depot it might not be the closest or the first available anymore.

The graph in fig. 4.1 displays an example of the behavior of Greedysize decrease.

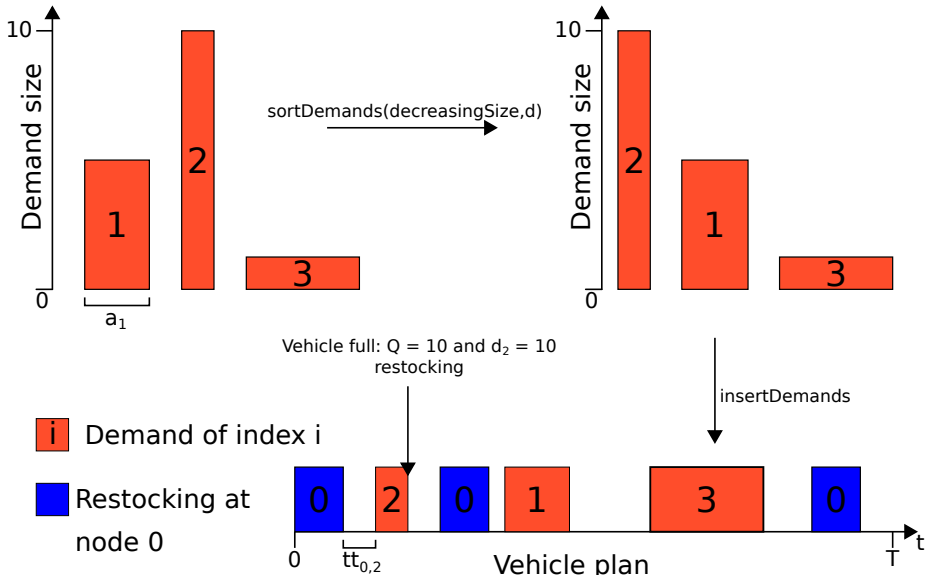


Figure 4.1: Example of Greedysize decrease for a vehicle of capacity of 10

In this example, we can observe how Greedysize decrease works on a simple case. Here, 3 demands need to be served using a single vehicle. In the second step, demands are sorted by decreasing size. Note that for the other greedy

heuristics, only the criterion for sorting demands is different. Then in the third step, demands are inserted in the route of the vehicle. The white spaces in the route represent the travel time between nodes. In this example, the considered vehicle has a capacity of 10. The rescue vehicle then has to go back to the depot after serving the first demand for restocking which means emptying the vehicle from the victims that are put to safety at the depot. Restocking is planned when a vehicle is full. Then it can continue its next route. This example only displays one vehicle. If several vehicles were used, the demand at nodes 1 and 3 would have been inserted in the route of another vehicle after the restocking was planned.

4.2.2 Solomon's heuristic

This heuristic has been mentioned in chapter 2. It is a reference in the literature introduced in [92]. The presented heuristic belongs to the category of insertion heuristics. In this thesis, the best performing heuristic from [92] has been selected. This heuristic starts by initializing every route according to a criterion. This criterion is selected among several proposed criteria presented in [92]:

- The farthest unrouted demand.
- The unrouted demand with the earliest deadline.

In this algorithm, 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 (the vehicle for which we arbitrarily plan the route first). Tours are incremented until all demands are served. We denote $h'^{z}_{j,k,u}$ the date of vehicle k at node j on tour z knowing insertion of demand u. The algorithm depends on parameters λ, 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, α_1 and α_2 are given in section section 4.4.

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

- The first criterion $c_1(i,u,j)$ is used to determine for each node u the best feasible insertion spot. $c_1(i,u,j)$ is computed for insertion of demand u between nodes i and j.

$$c_1(i,u,j) = \alpha_1 \cdot c_{11}(i,u,j) + \alpha_2 \cdot c_{12}(i,u,j) \quad \forall (i,j) \in \mathcal{V}^2, u \in \mathcal{V}^\star \quad (4.21)$$

With:

$$c_{11}(i,u,j) = tt_{i,u,c} + tt_{u,j,c} - tt_{i,j,c} \quad \forall (i,j) \in \mathcal{V}^2, u \in \mathcal{V}^\star, c \in \mathcal{C} \quad (4.22)$$

$c_{11}(i,u,j)$ is the deviation induced by insertion of u in the route. The new values of travel times to go to u from i ($tt_{i,u,c}$) and from u to j ($tt_{u,j,c}$) are summed. The former travel time from i directly to j is subtracted from this deviation ($tt_{i,j,c}$)

And:

$$c_{12}(i,u,j) = h^{'z}_{j,k,u} - h^{z}_{j,k} \quad \forall (i,j) \in \mathcal{V}^2, u \in \mathcal{V}^\star, k \in \mathcal{M}, z \in \mathcal{Z} \qquad (4.23)$$

$c_{12}(i,u,j)$ is the delay in the planning induced by this insertion. The new date of arrival to node j is subtracted by the old value to get the deviation. Contrary to c_{11} this criterion also takes into account the service time. Furthermore, in this adaptation of Solomon's heuristic, the factor μ usually put in factor of $tt_{i,j,c}$ in c_{11} formula has not been used. It is specified in [92]. In fact the value chosen for evaluation for μ is 1. That is why both c_{11} and c_{12} are used.

- With the first criterion c_1, the best insertion spot for each demand has been selected. The second criterion allows determining which demand is inserted knowing the results of best position selection with c_1. The demand selected is the one minimizing $c_2(i,u,j)$ with:

$$c_2(i,u,j) = \lambda \cdot tt_{0,u,c} - c_1(i,u,j) \quad \forall (i,j) \in \mathcal{V}^2, u \in \mathcal{V}^\star, c \in \mathcal{C} \qquad (4.24)$$

Solomon insertion heuristic pseudo-code is presented in algorithm 6.

4.2.3 Best Flow-time Insertion algorithms

Best Flow-time Insertion (BFI) is an insertion heuristic similar to the *best fit* allocation scheme from the bin packing problem literature. This algorithm has a performance ratio of 1.7 ([55]). It means that solutions computed by this algorithm are 1.7 times higher than the optimum in the worst case. This ratio is valid for the bin packing problem. In BFI the demands are sorted and inserted at the best position in the tours with the same principle of the Solomon heuristic. BFI sorts demands by priority and then by decreasing size, for demands with the same priority. The demands inserted first are the most likely to have the most impact on the objective. Then demands are inserted one by one into the routes. For each demand to insert, the algorithm tries each possible insertion position in the last route, for every vehicle. For each tested insertion, a score that presents the cost on the objective function (4.11) of the insertion is computed. Flow-time Insertion Score (FIS) is computed for the insertion in the route of vehicle k of node u after node i and before node j as so:

$$FIS(i,u,k) = \frac{p_u \times (tt_{i,u,c} + a_u + h^{z}_{i,k})}{q^{z}_{u,k}} + \sum_{j \in \Omega(i)} (h^{'z}_{j,k} - h^{z}_{j,k}), \quad \forall z \in \mathcal{Z}, c \in \mathcal{C} \qquad (4.25)$$

Algorithm 6
Solomon heuristic simplified algorithm

Input: d: list of demands
$k \leftarrow 1$
$z \leftarrow 1$
while $d \neq [\,]$ **do**
  $initializeRoute(k, d)$
  while $\sum_{i \in \mathcal{V}^\star} q^z_{i,k} < Q_k$ **do**
    $positions \leftarrow computeBestPositions(c_1, d)$
    $bestNode, position \leftarrow computeBestNode(c_2, d, positions)$
    $insertDemand(bestNode, position, d)$
  end while
  if $k = length(\mathcal{M})$ **then** ▷ All vehicles are full for the current tour
    $z \leftarrow z + 1$ ▷ Switch to next tour
    $k \leftarrow 1$ ▷ Select first vehicle
  else
    $k \leftarrow k + 1$ ▷ Switch to next vehicle
  end if
  $l \leftarrow sortDemands(criterion, d)$
end while

With:

$$\Omega(i) = \{j \in \mathcal{V} \quad | \quad h^z_{j,k} \geq h^z_{i,k}, \quad \forall k \in \mathcal{M}, z \in \mathcal{Z}\} \tag{4.26}$$

If the insertion leads to a deadline violation, the function returns $+\infty$. The simplified algorithm can be presented as in algorithm 7.

Algorithm 7 uses functions:

- *routeNodes*: This function computes the set of nodes in the route of a given vehicle. This is used to compute the predecessors to give to the FIS function.

- *insertDemand*: This procedure inserts the demand in the routes 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 Best Flow-time Insertion with Order Questioning (BFIOQ), no vehicle is a valid candidate for insertion without violating the deadline for the current demand. Then an error is logged to signal that no feasible solution has been found.

In fig. 4.2, the graph illustrates the principle of BFI on a simple case.

Algorithm 7
Best Flow-time Insertion algorithm

```
queue_of_demands ← sortDemands(d)
for cat ∈ categories do
    for dem ∈ queue_of_demands do
        bestScore ← ∞
        bestVehicle ← ∞
        for vehicle ∈ M do
            for predNode ∈ routeNodes(vehicle) do
                score ← FIS(predNode, dem, vehicle)
                if score < bestScore then
                    bestScore ← score
                    bestVehicle ← vehicle
                    bestPredecessor ← predNode
                end if
                if bestVehicle ≠ ∞ then
                    insertDemand(bestScore, bestVehicle, bestPredecessor)
                else
                    deadlineViolation(dem)        ▷ Handles deadline violations
                end if
            end for
        end for
    end for
end for
```

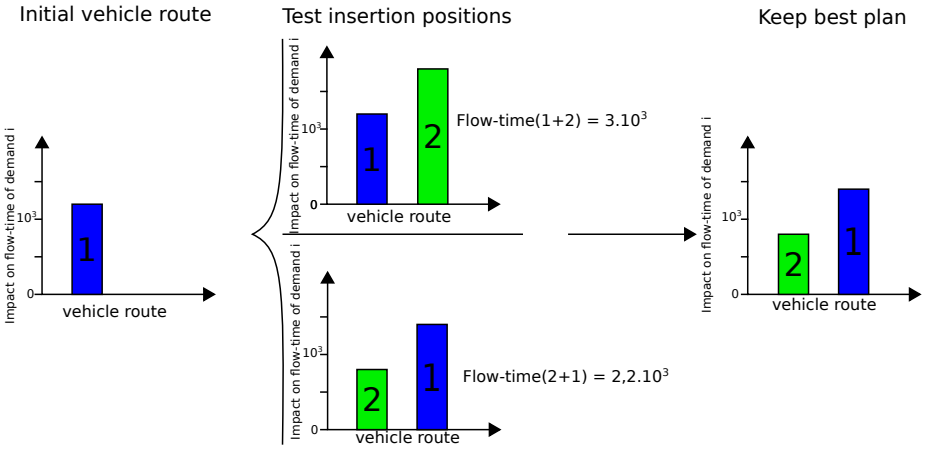


Figure 4.2: Example of insertion of demand at node 2 with BFI on a single vehicle

In this example, the main steps of the algorithm are displayed. To clarify the explanation, the figure only shows one vehicle, but the principle can be generalized to several vehicles as explained above. The first step is to test all possible insertion positions for the demand one wants to insert, for each available vehicle (here only 1). Then the score of each test is computed. The lowest score is kept as the best solution and the insertion of the demand is realized. If there are still demands to serve, the process is repeated until all demands are served entirely. In this example, we observe that the position of the insertion might also impact the Flowtime score of the demands that are already in the route, which has to be taken into account. It is the case in the tested insertion at the bottom where the impact of the first demand is increased but the benefit above all still makes this configuration the minimal in terms of objective score.

In fig. 4.3 another example is displayed with a single vehicle on a route that contains 2 served nodes.

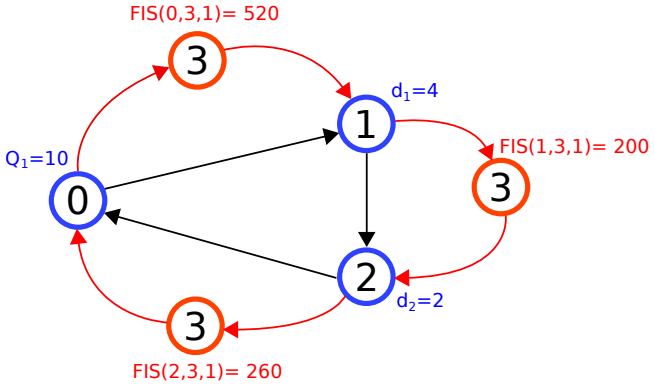


Figure 4.3: Example of insertion of demand at node 3 of quantity 4 with BFI on a single vehicle

In this example, the route already built for the vehicle is displayed as a graph. The FIS are computed for each insertion, and as the lowest FIS is for insertion between node 1 and 2, demand at node 3 is inserted at this position in the route.

Order Questioning

BFIOQ is only different from BFI from a routine launched after each insertion through a call to *insertDemand*. If the current route of the vehicle already contains at least two other interventions, 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.

The simple order questioning example in fig. 4.4 explains the principle with the insertion of demand in the route of a vehicle already containing 2 demands.

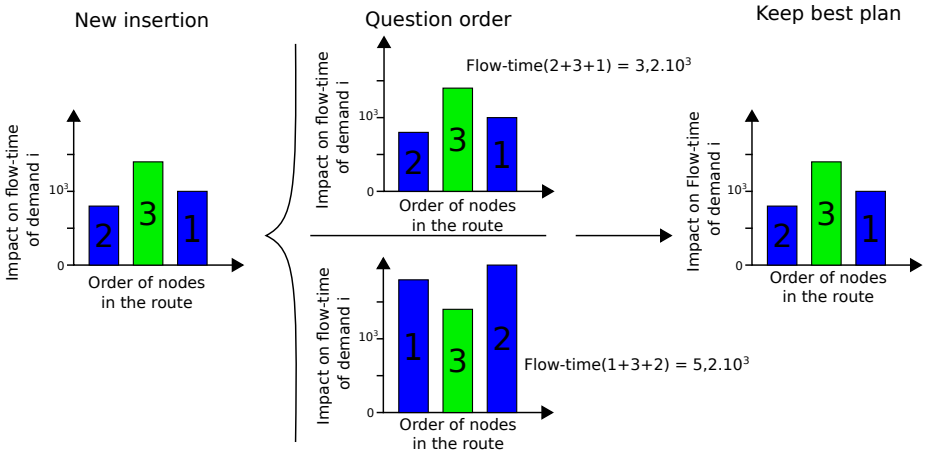


Figure 4.4: Example of Order Questioning with a single vehicle after insertion of a new demand on a route containing 2 nodes

Note that in this case, only 2 positions are tested, the current one and the one where the first and second demands are inverted. The configurations with the third demand at the first or last position have already been tried when doing the insertion process already described for BFI. The Order Questioning in this example did not manage to improve the solution.

4.3 Resources distribution problem

During a flood crisis, several separated areas are often impacted. In fact, for floods due to rivers, for example, different cities located near the bed are often impacted. When these cities are too far apart, several rescue centers are created and they might be handled as two separate problems. This problem is also raised when the flood cut a city in two for instance. Rescue teams often deal with such situations by creating a depot on both sides of the flooded area. However, even if there are two depot locations, it is often the same rescue team entity that needs to dispatch its resources among the depots. The problem of this section is to optimize the distribution of resources between sectors. Each sector is a connected graph with its own depot.

Three categories of resources dispatch algorithms are developed. The objective of these algorithms is to minimize the objective score sum over sectors one wants to dispatch resources on.

The first approach is to try all the possible configurations. For every possible resources distribution, a heuristic is executed on each sector and the obtained objective scores are summed. Then the best resources distribution is the one with the lowest sum of Flowtime objective score. It matches the lowest sum of objective scores obtained by the heuristic on the both graphs (fusion of the connected graphs). This algorithm is called Brute Force Resources Distribution

(BFRD). However, it is expected that a solution using brute force might be expensive in terms of computation time. Furthermore, since it uses a heuristic at every iteration, it is dependent on the computation time performance of the selected heuristic.

The second type of algorithm is a Greedy approach, Greedy Resources Distribution (GRD). This category of algorithms dispatches resources based on the ratio of a metric over the different sectors. Three GRDs are presented based on three different metrics:

- GRD1 dispatches to each sector the total number of vehicles multiplied by the ratio of nodes in the sector.
- GRD2 dispatches to each sector the total number of vehicles multiplied by the ratio of victims in the sector.
- GRD3 dispatches to each sector the total number of vehicles multiplied by the ratio of the sum of distances from the rescue center to each node on the sector.

GRD algorithms are faster to compute a resources distribution compared to BFRD. However, these algorithms have a higher risk of computing resources distributions which would lead to an infeasible problem.

The last category of algorithms is developed as a mix of the advantages of both previous types of algorithms. The goal is to try different configurations and evaluate them to give a good distribution. However, all possibilities will not be tested in order to try to improve computation time compared to BFRD. Constant Objective Detection Resources Distribution (CODRD) tries all configurations as well as BFRD. The global objective score on both sectors is computed for each distribution and resources are gradually reduced in the first sector while increased equally in the second one. The global objective decreases in the process until the minimum value is reached. Then, when we continue to transfer resources from the first sector to the other, the global objective increases. However, CODRD stops computation as soon as the global objective stops decreasing. If the global objective stagnates or increases, it is assumed that the local optimum distribution has been reached and there is no need to continue to transfer resources since the global objective of further distributions will be higher than the current value. For the experiments, we will apply these algorithms to 2 sectors but they can be generalized to more. To do so, we can consider two main graphs by merging sectors, and once the distribution is computed, we can zoom in on the sub-sectors recursively until a connected graph level is reached.

4.4 Evaluation

We study problems similar to the case study of Luchon. Therefore an analysis of Luchon Experience Feedback (EF) is conducted to extract values for the graph generation process. The algorithms of this chapter are evaluated on these

graphs generated with similar characteristics to the Luchon flood. The graph generator presented in chapter 3 is used to generate the graphs from extracted data.

4.4.1 Data analysis

To reproduce a real-life crisis to validate the model and heuristics, data has been extracted from EF of rescue teams of SDIS 31. 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. They can be adapted subsequently to discussions with the rescue teams to fit different situations.

In 2013 a flash flood has occurred in the valley of Luchon in the South Haute-Garonne department, France. The information that is contained in the EF documents is not sufficient to play the crisis identically. Nevertheless, data has been extracted to base experiments on. The objective is to build numerous graphs, similar to the Luchon crisis. These experimental graphs will be referred to as Luchon-like.

Data from EF concerns mostly interventions that have been logged. Information about these interventions is regrouped by category:

- **Category 1**: This category of intervention represents the 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 a high capacity, we set it to 30. During the crisis, this category represented 66% of victims rescued through 7 interventions.
- **Category 2**: 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 60 centimeters in water level in case of emergency. They have a limited capacity that 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. Relief operations are then operated by teams equipped with boats. These boats have limited capacity – set to 5 – for the experiments with 3 vehicles of this category covering the crisis. This category affected fewer victims: 8% of them dispatched on 15 nodes.

- **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 situations. 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 only human victims but also animals. Cattle can also be affected by a flood and it needs saving as well. Usually, 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.

The considered proportions are recapitulated below in table 4.3:

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%

Table 4.3: Category dispatching summary

The average speeds for each vehicle category are referenced in table 4.4:

Category 1	Category 2	Category 3	Category 4	Category 5
20 km/h	5 km/h	3 km/h	200 km/h	10 km/h

Table 4.4: Vehicle Speeds

Also the following values have been used for the graph generator in table 4.5:
Luchon-like graphs are completed using generated data. Values are then associated with these demands for the following variables:

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

Zone	1	2	3
Center	0,0	0,0	0,0
Size (in meters)	1000	2000	4000
Density (nodes per km^2)	10	5	1
Degree	3	2	1

Table 4.5: Graph generation parameters

- **Service time**: Uniform distribution on the interval $[\![5, 35]\!]$. This interval was given by the rescue teams from SDIS 31, the unit of time is the minute.
- **Demand size**: The law of distribution for these parameters depends on the category as stated in table 4.6.

In this table, the distribution normal law are truncated so that the support of the law is $[\![a, b]\!]$, *i.e.* the probability is null for a random variable to lie in an interval whose intersection with $[\![a, b]\!]$ is empty. The mean of the distribution is equal to μ and the standard deviation is equal to σ as illustrated in fig. 4.5. Using truncated normal law has been decided to avoid demands with too few or too many victims.

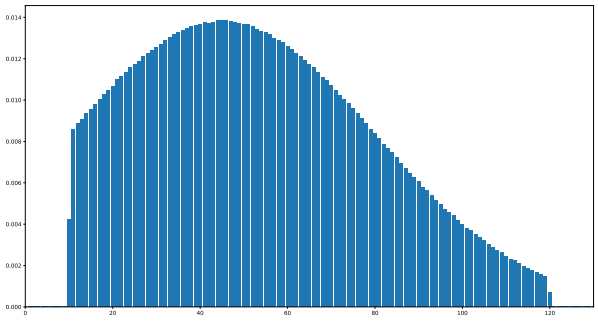


Figure 4.5: Truncated normal distribution with $\mu = 45$, $a = 10$, $b = 120$ and $\sigma = 35$

4.4.2 Small instances experiments

The evaluation process contains different experiments designed to compare the different forms of solutions presented for the static problem. The first experiment is conducted to evaluate the MILP. This evaluation aims at observing the scale of the problem upon which the computation time becomes too important to consider using this approach in an Emergency context. This evaluation cannot be conducted on Luchon's scale graphs since it is too big to obtain solutions

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 victim at each node	A single node of 30

Table 4.6: Demand size distribution laws by category

in a satisfying time. Smaller graphs are used for this experiment. Using the same set of graphs, an experiment is conducted to compare the greedy algorithm with BFI. The best heuristic of these heuristics is then compared with the MILP to ease the reading of the graphs by avoiding displaying all heuristics in the same figure.

In a second step, a comparison of the developed heuristics (BFI and BFIOQ) with other solutions (Solomon's heuristic from the literature and SDI as a baseline that models current rescue teams' behavior) is conducted on Luchon's scale graphs.

Exact method evaluation

This experiment has been conducted to evaluate the computational limits of an exact method with the MILP presented in section 4.1.1. It has been demonstrated in section 4.1.2 that the studied problem is NP-complete in the strong sense. Hence we expect an exponential computation time according to the problem size. To validate this assertion, an experimental set with graphs of increasing size is built. Small instances have been created from 2 to 12 nodes. Contrary to other experiments, the service times and size means have not been set Luchon-like. Smaller instances that can be solved with a MILP in a reasonable time are required. These inputs have been generated using a uniform distribution with different values. This process aims at observing the impact of the demand size mean and the service time means on the performance of the solution. The mean of service times and demand sizes have been drawn from the set $\{5, 20, 35\}$. These values are in minutes. For each triplet of demand size mean, service time mean, and graph size, 10 graphs are generated and tested. For this experiment, and since categories are independent, resources have been limited to only one category. For this category, 10 vehicles are available with a capacity of 10 each.

The MILP model, presented above, has been solved by GUROBI ([71]), a mathematical optimization solver that handles MILP resolution. It solves it using a branch and bound method. GUROBI solver under version 8.0.1 has been used on 8 parallel Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz Cores. Those are the performances of a laptop. This choice has been made because, in a crisis management context, the developed solution needs to be deployed in a field where computation resources might be limited. In addition, computation

time has been limited for the experiment. The limit of 3600 seconds has been chosen. One hour of computation is too much for the context and rescue teams want to start tours as soon as possible. It is considered that above 3600 seconds of computation time, the MILP solution is not relevant. When the computation time exceeds this limit, computation is stopped. If no feasible solution is found, an error is logged. If the solver managed to find a feasible solution but it might not be the optimal solution, it is logged as a sub-optimal result. The gap for this experiment has been set to 10^{-3}.

In fig. 4.6, the results of computation time of the MILP according to the graph size are displayed. The figure regroups different sub-figures, one for each combination of demand size and service time. It displays the percentage of graphs where computation led to an optimal result (in green), a sub-optimal result (in orange), or an error (no feasible solution in 1 hour in red). The axis titles are displayed on the sides of the figure and are the same for the axes of all sub-figures.

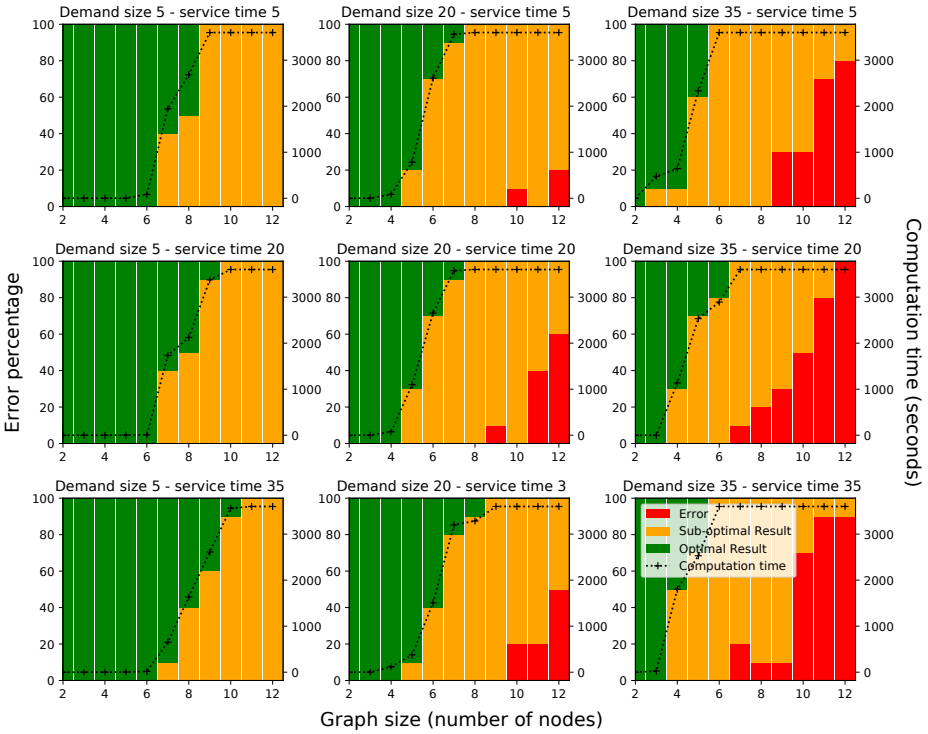


Figure 4.6: Computation time and *error rate* of MILP according to graph size

As a reminder, each point is the mean computed for the 10 graphs tested for each triplet of demand size mean, service time mean, and graph size. In this figure, we can observe that the computation time increases very fast starting with a certain size of the problem. The size where the breakpoint can be observed

depends on the constraints of the graphs (demand size mean and service time mean). For example, for a demand size mean of 5 and a service time mean of 5, the computation time exponential increase is problematic for the computation time requirements of our problem from problems of size 7. For a demand size mean of 35 and a service time mean of 35, the breakpoint appears on a lower size of problems (4 nodes). The first conclusion is that the model constraints influence the computation time. The more the model is constrained (more victims to rescue, more service time, ...), the earlier the computation time increases very fast with problem size.

Another observation is that, even in the less constrained cases, from 9 nodes problems and higher, no graph is solved with an optimal solution within an hour. It can be observed on the sub-figure in the top left corner where the sub-optimal solution's rate is at 100 % for graphs of size 9. For higher problem size values, and for the most constrained graphs, the error rate shows that 90% of the time, no feasible solution is found within an hour (graphs of size 12). However, this experiment does not allow telling if this is due to an infeasible problem or bad performances of the MILP.

Heuristics comparison

This experiment aims at comparing the heuristics presented above that have been developed for this work. The comparison is limited to greedy heuristics and BFI. The goal is to compare the greedy approach to BFI. This is why BFIOQ is not displayed and is to be evaluated in further experiments.

This experiment is conducted on the graphs used for the experiment above with the same resources. The same sub-figures are displayed. The graphics show the evolution of the objective score according to the graph size. The legend of the curves is only displayed in one sub-figure but is the same for all. The objective score is displayed, normalized by the objective score of the greedy heuristic GreedyAlea solutions. The results are displayed in fig. 4.7.

In this experiment, we can observe that, whatever the constraints on the problem (demand size and service time mean), the results are the same. BFI is the best of the compared algorithms. They all display better performances than GreedyAlea despite few exceptions. These results are as expected. In fact, since BFI questions previous insertions, it has more opportunities to improve its performances than greedy heuristics. GreedyAlea that selects demands and inserts them randomly is bad in several cases and could only show good performances in terms of solution quality on a few graphs by chance.

In terms of computation time, all algorithms compute a solution within a second. However, in some cases, no feasible solution has been found by the heuristics. In table 4.7 the error are displayed overall service time means and demand size means. The number of errors holds for the 990 graphs the evaluation is on. We may observe that the error rate is lower than the MILP except for GreedyAlea. Note that for fig. 4.7, graphs that generate an error have been removed from the graphics.

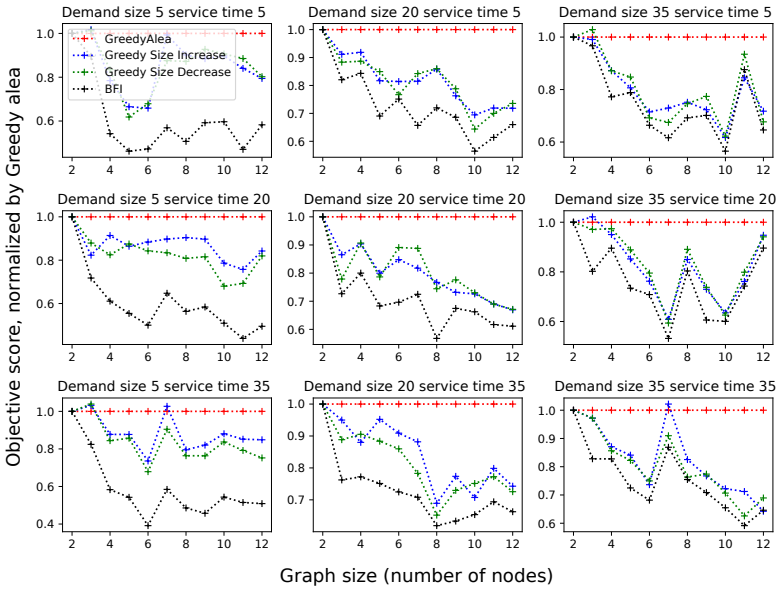


Figure 4.7: Objective score according to graph size of greedy heuristics and BFI

Greedyalea	Greedysize increase	Greedysize decrease	BFI
55	6	6	6

Table 4.7: Number of *errors* by heuristic over 990 Graphs

MILP and BFI comparison

The MILP reveals to be worse than heuristics in terms of computation time performances. BFI has shown the best quality of solutions in the previous experiment. This experiment evaluates the performances in terms of solutions quality of the MILP compared to BFI. The resilience to errors of each solution is also evaluated. The evaluation is conducted on the same graphs with the same resources as the above experiments. Results are displayed in fig. 4.8.

In this figure, as in the previous comparisons, we may observe that the constraints have an impact on the error rate of both solutions (BFI and the MILP). The error rate increases with the demand size mean and the service time mean. It also increases with the size of the problem.

The figure also shows that BFI's error rates are lower than the MILP's in any case and up to two times lower in the experiment where service time equals 35 and demand size 35 as well. Note that when a graph has led to an error for one of the solutions, it is removed from the comparison pool, and the mean is

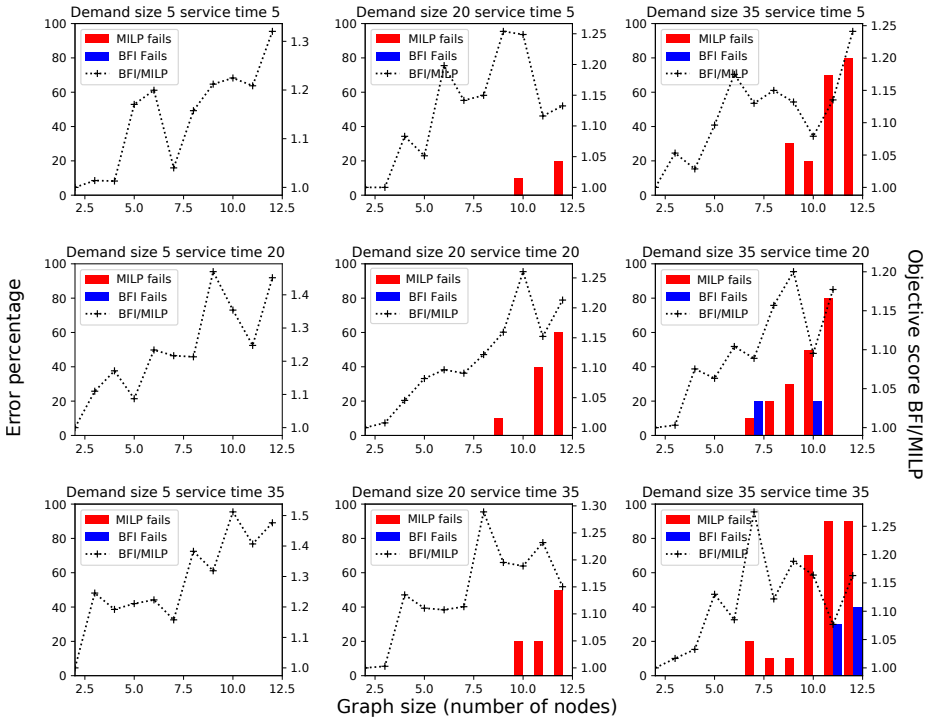


Figure 4.8: Objective BFI/MILP and errors according to problem size

computed on less data for this point of the graphic.

When the MILP finds a feasible solution, it is better than BFI even when the optimum has not been proved. But BFI is often lower than 1.25 times the objective score of the MILP and always less than 1.5 times. Furthermore, BFI computes under a second which could not compare to the computation time of several minutes up to the hour of the MILP. For these reasons, the MILP is not considered a suitable solution for our problem.

4.4.3 Luchon-Like experiment

For this experiment, graphs are generated with the Luchon-like characteristics mentioned above. The available resources to solve the problems are displayed in table 4.8.

Category	1	2	3	4	5
Number of vehicles	5	4	4	1	1
Capacity	30	10	5	1	10

Table 4.8: Available resources for the experiment, by Category

The objective of this experiment is to evaluate the performances in terms of computation time and solutions quality of the heuristics that we developed in this work and compare them to a baseline of the literature. In this experiment, and for the rest of the manuscript, the MILP is not used. Therefore, there is no motivation to use a linear objective function. The objective can be individualized so that every rescue is considered. This objective function is used to compute the objective score which will be the metric to compare solutions qualities in the rest of the manuscript:

$$\min \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{M}} \sum_{z \in \mathcal{Z}} (h^z_{i,k} - r_i) \cdot p_i \cdot q^z_{i,k} \tag{4.27}$$

Also, the graphs for this experiment (Luchon-like) are closer to a real-life crisis than the previous ones. Each graph counts a total of 500 victims over 60 nodes in total. The crisis is 12 hours long maximum (value of the highest deadline). The greedy heuristic that models the best the current behavior of rescue teams is SDI. This heuristic is evaluated in this experiment. BFI is selected as the best performing heuristics and its variation BFIOQ, which tries to improve the solution with local searches, is added to the comparison. Solomon's heuristic presented above is also evaluated in this experiment and is used as a baseline from the literature. The parameters used for Solomon heuristic (λ, α_1, α_2) are: (1,1,0), (1,0,1), (1,1,1), (2,1,0), (2,0,1) and (2,1,1). These parameters are suggested in [92]. For this last heuristic, the considered solution is only the best of the 6 runs, both for computation time and solution. It gives an advantage to Solomon's heuristic since the best parameters are used but only one computation time is considered and not the sum of the 6 runs.

The results of the experiments are displayed in fig. 4.9. This figure displays results by category. The objective score is accumulated from categories 1 to 5 from bottom to top. The computation time mean of the solution is displayed on the x-axis. For this experiment, the evaluation has been conducted on 100 Luchon-like Graphs and the results are the mean of these 100 results.

Results are recapitulated in table 4.9 with detailed figures. The deviations are given in regards to the heuristics that displays best Objective score performances, BFIOQ.

	Average Objective Score	Computation time (ms)	Objective Score deviation from best heuristic in Objective Score	Computation Time deviation from best heuristic in Objective Score
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 %

Table 4.9: Detailed Results of Heuristics Comparison

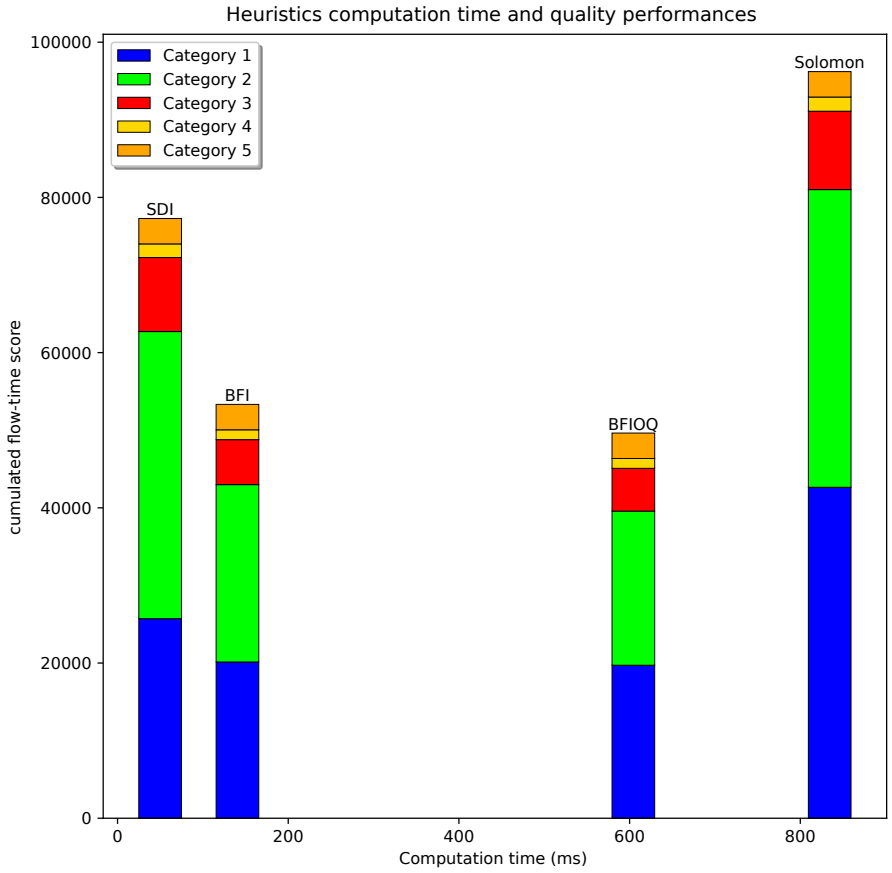


Figure 4.9: Heuristics performances in computation time and objective score

The results show that all heuristics developed for our problem are showing better performances both in terms of computation time and solutions quality than the selected baseline (Solomon's Heuristic). The fastest heuristic is SDI. The best solutions qualities are given by BFIOQ and BFI in this order. BFI offers the best compromise between computation time and solution quality. In fact BFI computes more than 4 times faster than BFIOQ and gives solutions only 7% higher than the best heuristic BFIOQ. BFI and SDI have a negative percentage for computation time since the comparison is made with the best heuristic in terms of Objective score (BFIOQ). However, the computation time performances are to be put into perspective since all heuristics compute solutions under the second. For this scale of the problem, all solutions tested fit the requirements in terms of computation time.

4.4.4 Resources distribution experiment

In this experiment, the different clustering solutions presented above in section 4.3 are evaluated. The Graphs are Luchon-like graphs that have been separated into two sub-graphs. The resources have also been changed a little. A vehicle has been added to category 4 in order to avoid sharing a resource between the two sectors. The results of this experiment are presented in fig. 4.10. In this graph, the results are presented in accumulated bars by category from 1 to 5 from bottom to top. The metric for evaluation is the objective score sum over both sectors of the solution found with Best Flow-time Insertion with Order Questioning heuristic (BFIOQ) with the dispatch computed by a given solution. The algorithms CODRD and BFRD are also using BFIOQ for their evaluation process. The number of errors (which is also a percentage since we did 100 runs) is displayed over the bar for each algorithm. An *error* signifies that the computed distribution did not allow BFIOQ to compute a feasible solution.

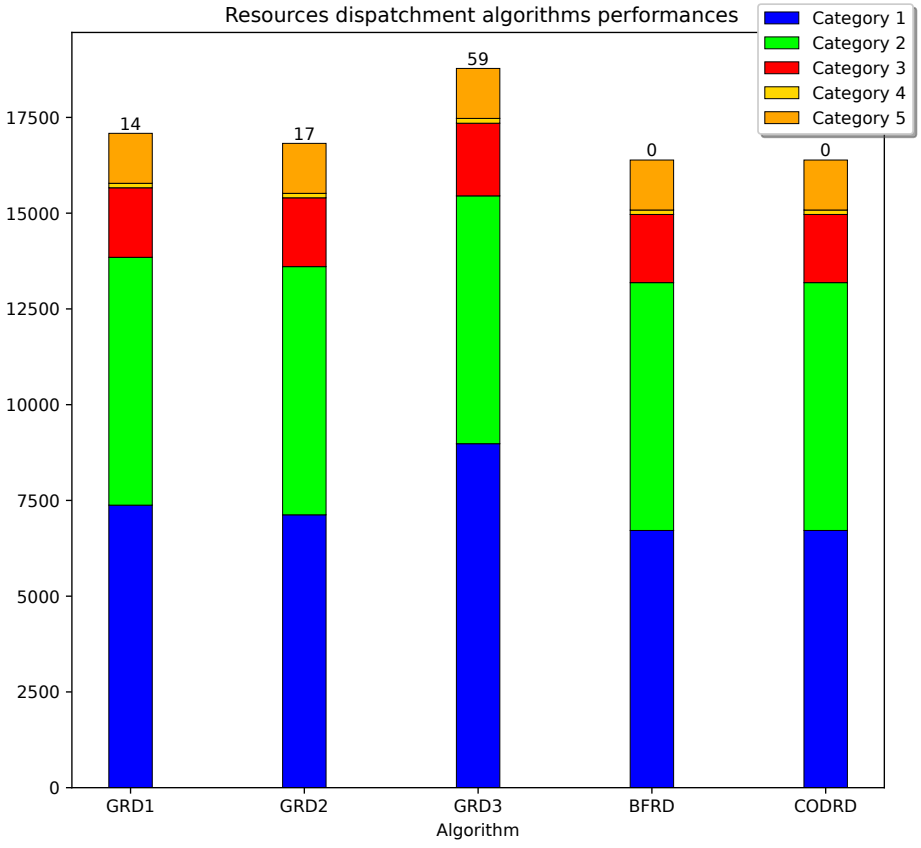


Figure 4.10: Resources distribution solutions evaluation with *error* percentage in top of the bars

In this figure, the results show that CODRD and BFRD perform better than the greedy heuristics. CODRD and BFRD give solution 4 % lower than the best greedy heuristic GRD2. BFRD and CODRD give the same distributions but CODRD computes faster, as expected. However greedy algorithms give a distribution almost instantly whereas CODRD takes up to 30 seconds for this experiment and is dependent on the size of the problem. Finally, the number of errors is to take into account. The greedy algorithms produce errors where BFRD and CODRD did not on the 100 clusters tested. GRD2 produces 17 % of errors, therefore CODRD is the most appropriate algorithm for our problem. It gives the best performances with the best stability. The computation time drawback is not important enough to disqualify it at this scale of problems.

4.5 Conclusion

In this chapter, the problem faced by rescue teams when dealing with flooding relief operations is presented. The studied problem is a static formulation where all information about the crisis is known at the beginning of the crisis. In this chapter:

- A MILP formulation has been presented and evaluated. It has been compared to different heuristics approaches.
- Greedy heuristics have been introduced.
- Insertion heuristics BFI and BFIOQ have been developed.
- A heuristic from the literature has been adapted to answer the characteristic of the problem.

The evaluation process has revealed that the MILP does not fit to answer the problem due to the computation time performances observed in such a solution. In fact for a scale of problems lower than the one we want to study (7 nodes), the computation time is higher than an hour, which is way too much for the crisis management context. These performances could have been improved by studying optimizations of the formulation or methods to guide the solver to find solutions faster, but it would probably not be sufficient to be fast enough for our problem's computation time requirements. The best performing heuristics are BFIOQ and BFI that insert demands in the route of the vehicle and at the position that has the lowest impact on the global objective score of the solution. As expected the local search on a turn, where the order of demands in a turn is questioned, embedded in BFIOQ, improves the solutions at the cost of computation time. These heuristics display better solutions quality than:

- Greedy heuristics based on demands sizes.
- A greedy heuristic SDI based on shortest distance and that models the current decision process of rescue teams.

- Solomon heuristic, an insertion heuristic from the literature.

Another problem that we studied in this chapter is the distribution of resources among different sectors. Different approaches have been presented and evaluated. The best performing one is CODRD, a clustering algorithm that tries all combinations of resources among sectors and stops when increasing resources does not improve the global score. The approaches have been evaluated in 2 sectors but the algorithms can be generalized to more sectors. However, with more sectors, CODRD could be too expensive in terms of computation time, and another approach that showed good performances, GRD2, could be helpful in such cases.

The problem presented in this chapter is static and could be used during the preparation phase of a crisis or for the purpose of exercises. However, during a real-life crisis, events append during the relief operations and need to be taken into account. Furthermore, information is dynamic and often not complete at the beginning of the crisis. A dynamic version of the problem is studied in the next chapter.

Chapter 5

Dynamic problem

The problem studied in the previous chapter is a static version of the real-life problem. It considers that all information about the crisis is known at its beginning. The solution has to be computed only once and the resulting route is followed without modification. As it has been reviewed in chapter 2, this formulation is perfectly adapted for some application fields. For instance, commercial VRP applications could be solved with a static approach because few unexpected events might occur. During a catastrophic event's relief operations phase, the complete situation of the field is rarely known at the beginning of the crisis. Information has to be gathered in a degraded environment and data is communicated to the rescue teams continuously. Floods are even more specific in that matter than other disastrous events. Contrary to catastrophes such as earthquakes where the source of the damage lasts for a short period of time, for flooding events, the situation evolves constantly because water levels are still varying during the rescue teams' interventions. In this context, not only is the information likely to be released during interventions, but the situation of known demands also evolves. The category of VRP that encompasses these issues is DVRP. This chapter studies the encountered characteristics when dealing with a real-life problem of people relief operations in the context of flash floods. First, in section 5.1, the dynamic problem is described and compared to the static version. A description of the dynamic sources of events is presented and put into perspective with the literature. The mechanisms and algorithms studied to optimize rescue vehicles' routing during the real-life victims' relief operations are presented in section 5.2. Finally, section 5.3 presents the followed methodology to build the experimental sets, the experiments that were conducted, and the analysis of the results.

5.1 Problem description

In this section, the dynamic version of the problem is presented. In this problem, the response phase is studied. In fact, in the static problem, all demands

were known at the beginning and were not subject to evolution. Here, the studied problem corresponds to the response phase of the relief operations. It is considered that few anticipation operations are possible and routes need to be updated dynamically to respond to the evolution of the situation. Some of the hypotheses of the static problem need to be modified and others need to be added:

- Travel times and service times are still considered to be known but in this problem, delays are possible on these values. Information evolves dynamically, hence dynamic events may affect the operations. These events might delay vehicles from their routes and therefore need to be incorporated into the model.
- Release dates are the date at which information is known. For the static problem, information was entirely known at the beginning of the crisis. Release dates have been used as the beginning of the time window (the period during which demands need to be served). In the dynamic problem, events are released during the crisis. The release date is then the date when the event appears in the problem.
- Vehicles cannot be re-routed. Routes are updated dynamically during the crisis to adapt to the situation. However, at least in SDIS 31, there exists a rule stipulating that a departed vehicle, moving to serve a demand point, cannot be re-routed. It is a stability criterion required by rescue teams' operation process. A route can be modified only after the end of service at the demand it is driving to.

5.1.1 Dynamic events

In this work the dynamic events that we consider are:

- **Dynamic demands**
- **Delays** on travel or service
- **Updates** of quantities, priorities, travel time or service time
- **Dynamic resources**

As mentioned in section 2.2.1, different types of dynamic events are studied in the literature. These types of events are not exhaustive but are the most spread in DVRP. The considered dynamic events are detailed below and associated with the dynamic events from the literature:

- **Dynamic requests:** Requests are revealed dynamically in the problem. In fact, at the beginning of the crisis, a set of issues are known. These issues are generally places that host public such as hospitals, retirement homes, schools, hostels, or enterprises. Rescue teams have databases about

such issues and in correlation with the time of the day they can approximate the number of victims at these locations. For example, if the crisis happens during the day, enterprises and schools will become issues with many potential victims whereas during the night these places are empty and do not require to be handled. These types of issues often host numerous people and are considered at the beginning of the crisis because it may be very difficult to evacuate such locations if the water rises too high and a lot of victims have to be evacuated. Furthermore, places like hospitals or retirements homes host fragile people, harder to rescue with a high water level. The rest of the demands are not known or uncertain with no probability information in our case. Some people might evacuate before it is too late and never need help from rescue teams. Others might want to stay at home and when the situation gets worse, they will need to be rescued. This type of issue is revealed during the crisis in two main ways. First of all, at the beginning of the crisis, rescue teams start a scouting process with dedicated resources. Small vehicles travel the impacted territory to gather information about the situation on the field and report back to the rescue center. Rescue teams also gather data thanks to calls from victims. Most of the time they call either for guidance or for help. The rescue teams estimate if an intervention is necessary, and add the demand to the problem if so. When a node is added, it is recorded with all its characteristics described by the inputs including its release date with the current date.

- **Dynamic demands:** As a reminder, the term **customer** is often used instead of **Demand** in the literature where most of the VRP studied have commercial applications. As mentioned for the dynamic requests, when a new demand is released, the node is added to the problem with all its characteristics. However, these inputs are estimated. For example, the service time at the node is an estimation made by the rescue teams in consideration of the node characteristics. On the same principle, victims' state of health is estimated through phone calls but might be different in reality and therefore the priority of victims as well. Finally, the number of victims actually to be rescued at a node might evolve. Victims from the same neighborhood may regroup for instance. In these cases, the quantity of the node where victims gather is increased. A node of the problem might also be deleted since victims are not at the location anymore. Once at the node, rescue teams know with certainty the actual situation. In case inputs are not pertinent, two types of events have been defined.

  Delays enable to signal lateness or more rarely an advance on the planning. Only service delays correspond to dynamic demand events from the literature. This type of events signals that the vehicle that communicated it will not be able to respect its planning. We define a delay as an unexpected event during service that might not be reproduced next time. For example, if one of the victims is injured and cannot walk. Rescue teams take then more time to rescue this victim, but it does not impact the next

intervention at the node and the graph is not modified.

To signal a permanent modification of an input, **Updates** are used. This type of event is used to report permanent changes in a priority, quantity, or service time. It may be due to an evolution of the water level at the node. If the water level is higher, it may take longer to evacuate victims. Then, service time needs to be updated.

- **Dynamic travel times:** The travel times are also dynamic in the problem. They are signaled through both types of events introduced for dynamic demands. **Delays** on travel time may be emitted to signal lateness on a route between two nodes that may not be permanent. For instance, if a tree falls on a road, the rescue teams will have to resolve the situation to get through but the operation only affects the first vehicle using the route. One does not want to modify the graph but needs to signal the lateness to the rescue center to adapt the routes accordingly. **Updates** on travel time on the contrary occurs when a road segment is completely blocked for example. Therefore, a detour might be necessary and the graph is modified.

Dynamic Resources is the last type of dynamic event considered in the problem. The resources in the problem are the rescue vehicles. As mentioned in the static chapter, a flooding crisis often affects several areas at the same time. Resources may be shared and the pool of available vehicles for each category is subject to evolution. A resource can either be added to the pool of vehicles or removed.

5.1.2 Variables

Characteristics of static and dynamic versions of the problem are very similar. All variables and inputs of the model that were introduced in the static version are still relevant with the dynamic version. The table 5.1 offers a reminder for variables and table 5.2 for inputs:

$x^z_{i,j,k}$	Binary variable equal to 1 if vehicle k uses 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 rescued by vehicle k at node i on tour z

Table 5.1: Variables

The main difference with the static formulation concerns the release dates. In fact in the static problem, they were considered known from the start, and in the experimental part, set to 0. It has been considered that rescue teams do not wait until a certain date to serve a node but try to deal with the demand as soon as it is revealed. In the dynamic formulation, the time window is not left-bounded. Demands are to be treated as soon as they are released. However, release dates are non-null because events and demands are revealed dynamically.

p_i	Priority: A coefficient used in the objective function for demand i
f_i	Deadline: latest time for any vehicle to serve demand at node i entirely
r_i	Release date: time when the demand i appears in the problem
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	Service 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 5.2: Inputs

Hence release dates in this problem also play the role of reveal dates: the date at which the information about the demand is available for the rescue teams. It seems important to make focus on that aspect since most of the time, in the literature, release dates are used differently. In scheduling or the VRPTW literature, release dates are mostly used to define the date when the service can begin. This difference is mostly due to the crisis management context where demands can be treated as soon as they are revealed, and there is no need to wait for a starting date as we would for delivery to a demand for instance. This is why the problem has been qualified as a Vehicle Routing Problem under Deadlines (VRPD) instead of a VRPTW.
Another set of variables is defined to cover the dynamic problem. During a crisis, events are released dynamically. Rescue teams must update the routes to take into account this dynamic information. But these new routes are computed while vehicles are already serving demands. The state of the vehicle fleet needs to be considered. Information is required for each vehicle:

- The node it is currently serving is needed. Rescue teams do not want to be re-routed. This means that when a vehicle departs to serve a node, the routes cannot be changed before the date of the end of service at this demand point as illustrated in fig. 5.1. This figure displays the planned route for a vehicle on a tour. The red route is the route to the current node, which cannot be modified. The rest of the route in green can be questioned dynamically. The route can only be updated after the current node. We note the current node:

$$nc_k, \quad nc_k \in \mathcal{V}, \quad \forall k \in \mathcal{M} \tag{5.1}$$

  It either describes the node being served by vehicle k, or the node vehicle k is driving to.

- A given node might appear in several missions of the same vehicle, on different turns of the routes. Let us take the example of a vehicle serving a node, driving back to the depot and going back to this same node to rescue the rest of the victims. If only the current node was available, there is no way to know if we are currently serving it for the first time or the second time. Coupled with the current turn, one can know exactly which mission from the routes are being served. The current turn in the routes is:

$$zc_k, \quad \forall k \in \mathcal{M} \tag{5.2}$$

- To update the routes dynamically, the available date for each vehicle is necessary. It allows planning of missions for a vehicle only after its available date. Missions cannot be planned before because re-routing is not allowed. This date corresponds to the end of its service at the current node. We note this date:

$$hc_k = h^{zc_k}_{nc_k,k} + a_{nc_k}, \quad \forall k \in \mathcal{M} \tag{5.3}$$

  This variable could be computed from the routes every time it is needed, but it would be more expensive in terms of computation time. The variable is updated dynamically instead.

- To plan missions, the current number of victims in each rescue vehicle is required. With this information the number of victims that can be rescued by the vehicle in the current turn can be deduced. Therefore it allows planning of missions for the current turn of the vehicle according to left space. This variable, updated every time victims are rescued by a vehicle, is called the load:

$$qc_k = \sum_{i \in M^k_{zc}} q^{zc_k}_{i,k} \quad \forall k \in \mathcal{M} \tag{5.4}$$

  With

$$M^k_{zc} = \{i \in \mathcal{V}^\star, h^{zc_k}_{i,k} \leq hc_k\} \quad \forall k \in \mathcal{M} \tag{5.5}$$

A reminder of these variables is given in table 5.3:

nc_k	Current node of the vehicle k
zc_k	Current turn of the vehicle k
hc_k	Available date for vehicle k
qc_k	Current load of the vehicle k

Table 5.3: Vehicle state variables

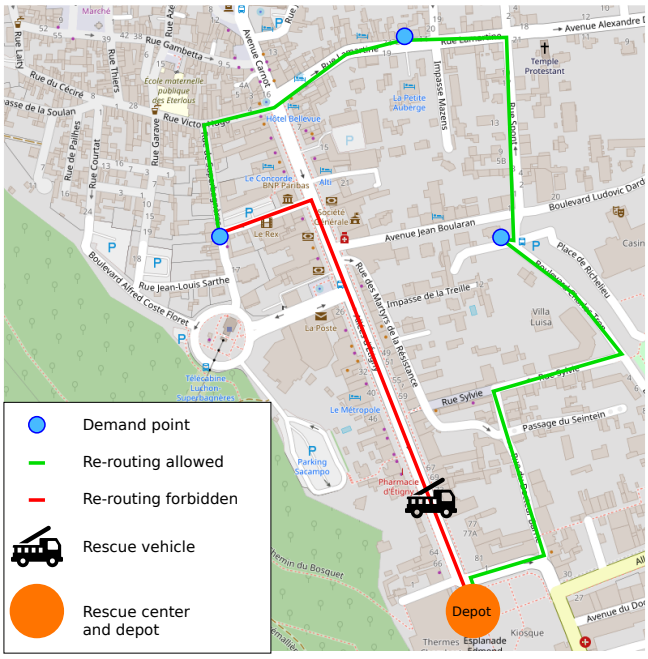


Figure 5.1: Example of forbidden re-routing on a tour of a rescue vehicle

5.2 Solutions for the dynamic Vehicle Routing Problem

Solutions to a dynamic problem have to be updated regularly to consider the dynamic events and adapt routes accordingly. Routes updates are often realized by dynamic heuristics. In fact, in a dynamic context, exact methods are often too expensive in terms of computation time. The challenge is to have a completion time for the solution which is comparable to the values of inter-arrival times between dynamic events. If not, several dynamic events with major impact could be released during the *routes computation*, and opportunities for improvement could be missed. In the worst case, it could even lead to an infeasible problem if a dynamic event is handled too late. Considering the static experimental results with exact methods, the option of using heuristics seems the most appropriate. The heuristics that we develop to tackle this dynamic problem are presented in section 5.2.2. But whatever the method of solving, some specific issues to the DVRP have to be considered. The first issue is to decide when to update routes dynamically. Then, the conflicts between the vehicles that execute the routes on one side and the DC that updates the routes on the other side, need to be solved. A solution to the dynamic problem is the combination of a *routes computation heuristic* and a *routes computation strategy* concerning a Computation Period.

5.2.1 Dynamic mechanisms

Solving a dynamic problem raises several challenges. On the one hand, there is concurrent access to the routes between vehicles serving their missions and the DC that tries to update the routes. The DC 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 and missions that are already being served should not be moved later in the routes. On the other hand, a choice has to be made about when to launch *routes computation*. *Routes computation* defines the operations conducted at the decision center to update the routes of operations dynamically. The mechanisms that handle these challenges are presented in the next sections.

Routes computation period

When *routes computation* is started, a neutralized period called the Routes Computation Period (RCP) is defined from the current date. Routes included in this period are immune to changes. It allows ensuring that during *Routes Computation*, a vehicle will not serve a mission that is outside the RCP without DC considering its new state. If *routes computation* is still running when the end of this period is reached, it is interrupted and the length of this neutralized period is doubled. Only after, the *routes computation* is relaunched. In fact, when this period is over, the DC could be questioning missions that are being served in the meantime. For instance, it could be decided to put the mission later in the routes making these new routes obsolete.

The initial value for RCP is determined by experimentation. Figure 5.2 presents an example where the principle of the RCP is illustrated on a single vehicle with *routes computation*. The RCP is illustrated through the second *routes computation*, the computation time reached the RCP value and it is not guaranteed that we would not question the demands that are being served hence the *routes computation* is interrupted. Indeed, the figure shows that if we continue computing after the orange area, the *routes computation* may last until the date of the next mission. Also, in the third *routes computation*, the RCP is doubled after the interruption of the previous *routes computation*. This is called an **overtime**. This *routes computation* shows a fixed (not questioned) mission because its service is included in the RCP. It also displays a mission later in the routes that have been offset.

At the beginning of a *routes computation*, a copy of the state is passed to the *routes computation heuristic*. It is updated to represent the state of each vehicle at the end of the RCP.

Routes computation strategies

A second mechanism is necessary to decide when to start a new *routes computation*. Three *routes computation strategies* have been developed. The first mechanism, RCP is common to all the three *routes computation strategies*. For all presented *routes computation strategies*, dynamic events are stored in a buffer

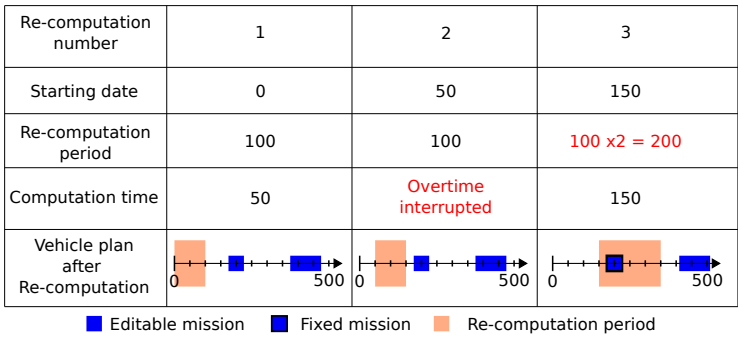


Figure 5.2: RCP example

when waiting to be handled in *routes computation*. The *routes computation strategies* specify when the *routes computation* is launched and when a dynamic event is added to the buffer.

- **Preemptive**: *routes computation* is launched at every dynamic event. When a new event is released, *routes computation* is interrupted. Events of the previous *routes computation* are kept in the buffer. The new event is added and a new *routes computation* is launched.
- **Continuous**: When dynamic events are released during a *routes computation*, they are buffered. *routes computation* is re-launched as soon as the previous *routes computation* ended if the buffer is not empty. If it is empty, the next dynamic event triggers a *routes computation* with only this event.
- **Periodic**: *routes computation* is launched at **RCP** time interval with all the dynamic events since the previous *routes computation*. Contrary to the continuous strategy, once a *routes computation* is over, it waits for the end of the RCP period and events that have been released during this period are buffered.

Figure 5.3 illustrates how the different *routes computation strategies* work showing when *routes computation* is started according to releases of dynamic events. In this figure, the **Overtime** process is not illustrated to clarify the explanations, but note that it does not affect how these *routes computation strategies* work. The buffers that have been displayed show the events that are taken into account in the *routes computation* under it. Therefore when an event disappears from the buffer, this means that it has been integrated into the routes.

5.2.2 Routes computation heuristics

The section 2.2.1 presents various approaches to solve a DVRP. Due to the dynamic nature of the problem, the computation time for new solutions needs to

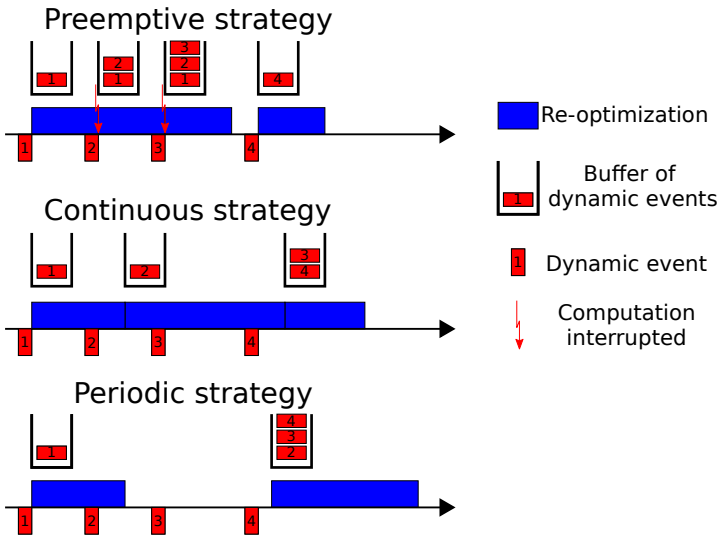


Figure 5.3: Dynamic *routes computation strategies*

be kept within reasonable bounds to avoid leading to an infeasible problem. The static study has already enabled to observe computation time performances of an exact method. It has revealed that for graphs of 12 nodes, the *routes computation* took more than an hour on standard inputs. The problem is that scenarios that need to be treated are bigger than 12 nodes graphs. This justifies that exact methods do not fit to answer this dynamic problem. Other solutions have been presented in the state of the art. The stochastic approach is often used to solve DVRP as in [16], [97] or [22]. However, this approach is based on probabilistic information. In the context of crisis management and more specifically in this study, it would be very ambitious to hope to gather such data. If one considers the example of dynamic demands, every housing infrastructure is a potential issue that could become a new demand. But no reliable probability exists on the chances it actually becomes one. Therefore, the stochastic approach is not adapted to this problem either.

Re-optimization approach is often used in the literature for problems where computation time is a major challenge for the solution. Since heuristics have been developed for the static formulation of the problem, the re-optimization approach can be coupled with the best static heuristics to solve the dynamic problem.

The literature also offers insertion heuristics. These heuristics are suited for dynamic demands problems. The principle is to insert the new demand in the existing routes, contrary to the re-optimization approach, where the routes are built from scratch at each *routes computation*. It allows saving computation time and it would be interesting to observe whether this advantage copes with the potential loss in solution qualities.

Finally, a *local search* heuristic is presented in this section. These solutions are used to try to improve the routes between *routes computations*. The solutions that are presented in this section correspond to the *update_routes* method mentioned in section 3.2.1. The routine *update_graph* that updates the graph information from the received events is also embedded in the developed *routes computation heuristics*.

Re-optimization approach

The Re-optimization approach is widely spread in the literature. This approach uses a static heuristic and re-computes the routes dynamically with it. The basis of the approach is to run an exact solution but the semantics has been enlarged to include heuristics. In this work, the static heuristics that gave the best results in the static analysis are selected. BFI and BFIOQ will be used. These heuristics have been presented in section 4.2.3.

However, some mechanisms are added to these *routes computation heuristics* in order to be run dynamically.

First, deadline violations are considered. In case the *routes computation heuristics* did not manage to find a feasible solution, once a deadline violation is detected, the *routes computation heuristic* is interrupted. In fact, if no insertion position avoids deadline violation, it mainly means that a bad decision has been made earlier during the routes building. Therefore, an Earliest Deadline First (EDF) algorithm is used to find a solution that could avoid deadline violations. This process is used in the dynamic problem and not in the Static chapter. In fact in the static formulation, we only wanted to analyze when the Route Computation Heuristic managed to find a feasible solution. In the dynamic problem, if a feasible solution is not found during a *routes computation*, it impacts the rest of the operations. EDF is a scheduling algorithm introduced by ([67]) under the name Deadline Driven algorithm. It inserts demand into the schedule (here being the routes) by deadline order. In the current problem, several vehicles are concerned. When the demand to insert is selected for its deadline, a choice has to be made about which vehicle will deal with this demand. Since the algorithm is driven by deadlines, the best choice is to select the first available vehicle in the current routes. The first available vehicle is the one that can arrive at the node first after serving previous nodes from its route. If EDF finds a feasible solution, then it is returned. In the other case, the *routes computation heuristic* failed to find a feasible solution and an error is logged. An error signifies that no satisfying solution has not been found and the manual intervention of rescue teams is necessary to serve the demand that led to deadline violation.

Another mechanism is added to the *routes computation heuristics* to handle the current state of vehicles. Variables used to record vehicles' state have been described above. When *routes computation* is launched, the current state of each vehicle is copied and updated with values that translate the state it would be in at the end of RCP. Then the *routes computation heuristic* loads this state and creates initial routes that cannot be modified. These initial routes correspond

to the routes being currently served during *routes computation*. These initial routes contain the last demands to be served and load the vehicles with victims.

One last thing to consider is how the different types of dynamic events are handled with this approach.

- **Dynamic Demands**: They are added to the pull of demands to be inserted, which is an input for the *routes computation heuristics*.
- **Updates**: This type of event modifies information about the graph. When they are received, the graph is updated and the new field situation is taken into account with the next *routes computation*.
- **Delays**: The timing of the routes is affected and the state is modified in order to include this information for the next *routes computation*.
- **Dynamic Resources**: Variables that describe the pool of available vehicles are updated in order to add or remove a resource.

Insertion heuristic

An insertion heuristic may be used to deal with dynamic demand problems. This heuristic is inspired by the dynamic neighborhood search approach that can be found in [44]. This heuristic only inserts the new demand in the existing routes instead of recomputing the routes entirely. However, the main drawback is that this type of heuristics is adapted to answer dynamic demands problems. Hence, the other types of events from the problem will have to be handled separately. A strategy from ([78]) has been studied. It presents an algorithm that inserts demands at the end of the route when they are released. The solution has been adapted and improved to insert demands at a different position in the routes, not only at the end. If insertion is only made at the end of the route, and since the studied problem work with deadlines, the insertion at the end of the route could lead to deadline violations.

This insertion heuristic Flow-time Insertion Phase-out (FIP) uses an adaptation of the FIS presented in section 4.2.3. The principle is to find a mission in the routes to swap with the demand to insert. To select this mission, the impact on the objective score of the insertion is computed with Flow-time Insertion Phase-out Score (FIPS). The goal is to find a mission that could be exchanged with the demand to insert and that would have a positive impact on the objective score.

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 FIPS is expressed as follows:

$$FIPS(i,m,k,z) = p_i \cdot h^z_{i,k} - p_{m.node} \cdot h^z_{m.node,k} + \sum_{n \in nextNodes} p_n \times (h^z_{i,k} - h^z_{m.node,k}), \tag{5.6}$$

where *nextNodes* is the list of missions from the routes positioned after the phased-out mission. Note that the quantity is not part of the score because

insertions of demands in the routes are realized by exchanging with a mission already in the route. Hence the quantity removed from the mission in the route is equal to the quantity of the demand inserted. If the demand to insert is bigger or equal to the mission's quantity, we only insert the quantity of the mission. In the other case, if the demand is smaller than the mission's quantity, the demand is inserted entirely and the difference in the quantity of the mission is kept in the route.

This score takes into account the cost of removing the mission from the routes as well as the cost of inserting the new demand. This way, it ensures that benefits from insertion are worth phasing out the mission. The objective is then to lower the FIPS. The mission with the lowest negative FIPS is selected and swapped. The phased-out mission is placed in a queue of demands. This queue is sorted 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 routes 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 in algorithm 8.

Algorithm 8
Dynamic insertion algorithm

```
queue_of_demands ← newDemands
while queue_of_demands ∈ categories do
    i ← queue_of_demands[0]
    list_of_missions ← computeListOfMissions()
    bestScore ← 0
    while q_i ≠ 0 do
        for m ∈ list_of_missions do
            if FIPS(i, m, m.vehicle, m.turn) < 0 then
                bestScore ← FIPS(i, m, m.vehicle, m.turn)
                bestMission ← m
            end if
        end for
        if bestScore ≠ 0 then
            insertMission(bestMission)
        else
            insertAtTheEnd(m)
        end if
        list_of_missions ← computeListOfMissions()
    end while
end while
```

Algorithm 8 uses functions:

- *computeListOfMissions*: Computes the missions from the routes and returns a list of missions, candidates to be phased-out. Note that this function takes into account the state described in section 5.1.2. The missions from the routes that are under the RCP are not added to the list because one does not want them to be candidates for phasing-out.

- *insertMission*: Replaces a mission in the routes and adds the phased-out mission in the queue of demands. The quantity of the demand left to insert is updated if the demand to insert is bigger than the phased-out mission. If the phased-out mission is bigger than the quantity to insert, the demand rest of the mission is not phased-out and is kept in the routes.

- *insertAtTheEnd*: Selects the first available vehicle in the current routes and inserts the demand to complete the last turn. If the demand is not entirely inserted, the quantity left to insert is updated, else the demand is removed from the queue.

- The fields of the mission m accessed like $m.vehicle$ are presented in section 3.2.2.

In fig. 5.4, an example of FIP is illustrated. In this example, there is only one vehicle. Demand needs to be inserted (demand to node 3). A vehicle route in this diagram is illustrated with the missions services on the timeline as well as their impact on the objective score on the y-axis. The route of the vehicle already contains missions to nodes 1 and 2. In the right column is displayed the total objective score of the routes. In the first step, the route before insertion is shown, and the queue is filled with the demand to insert. After testing the different insertion positions, the choice is made to insert the demand to node 3 in the first position and to phase out the mission to node 1. It means that the mission to node 1 is removed from the route of the vehicle and put in the pool of demands. Mission to node 3 is inserted instead with the same quantity. This decision is made according to the FIPS and is illustrated through the arrows. In step 2, right below step 1, one can observe that a benefit has been obtained by this insertion (the total objective score has decreased). The mission to node 3 represents a lower impact on the objective than the previous mission at this position. Furthermore, the travel time (space on the timeline between two missions), between node 3 and node 2 is lower than between 1 and 2. Hence a gain on the objective is also reached on the mission to node 2 (served earlier than before the insertion). After testing all insertion positions, the decision is made to insert demand at node 1 at the end of the route as illustrated in steps 2 to 3. The impact of the mission to node 1 is a bit higher than before insertion, however, the total impact of all the insertions is lower than if we just inserted the demand at node 3 directly at the end of the route.

Note that, in this example, the case is simple because demands of the same quantity are considered. Therefore, there is no need to split demands to insert it.

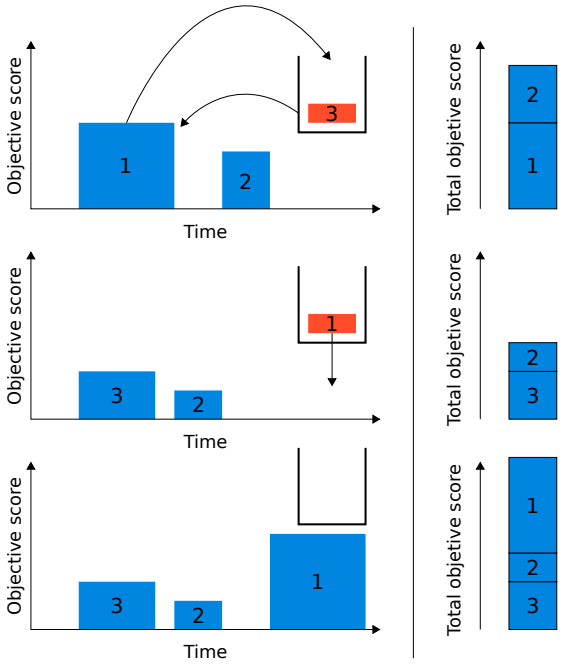


Figure 5.4: Example of FIP usage

In case a deadline violation is detected during an insertion (no insertion candidate offers negative FIPS without deadline violation), the score evaluation/insertion routine is re-launched. However, in this routine, the goal is not to find a negative FIPS but just to select the best insertion position that does not violate deadlines. If after this routine, no valid candidate is found, the *routes computation heuristic* returns a deadline violation error. In this case, EDF is not used since we do not want to mix the re-optimization approach and the insertion heuristic. EDF is an algorithm that computes a solution from scratch in opposition to the insertion heuristic. Using EDF in this context would introduce a re-optimization approach in the insertion heuristic where the objective is to compare both approaches. When a deadline violation is detected, the same insertion algorithm is launched but instead of trying to find a negative FIPS (which translates a benefit on the global objective score), we only insert at the position where the insertion gives the lowest FIPS. If after this process, there is still a deadline violation detected, an error is logged to signal it.

The problem of this heuristic, in opposition to the re-optimization approach, is that it only deals with new demands (dynamic demands). For the other types of events, different processes need to be specified:

- **Updates**: The graph is updated with new information. This information will not be included before the next new demand. The routes are updated according to the type of update. For example, for an update on travel

time, the planning is updated accordingly with the new travel time value. Routes are not computed all over again.

- **Delays**: Routes are updated considering the delay on the impacted vehicle. The rest of its missions are offset accordingly.
- **Resources Modification**: If resources are added, the vehicle is pushed into the pool and it will be considered available for the next insertion. In case of resource removal, routes for the vehicle are deleted and all the missions are added to a queue of demands. An insertion is launched with all these demands to insert.

During ingestion of these events, a deadline violation might be produced. In this case, it is detected and logged as an error. There is no process to try to recover from these errors for the same reasons mentioned about EDF, we do not want to mix approaches. However, for the purpose of a final solution, the approaches could be mixed to improve performance. As observed, since the routes are not questioned for these types of events, it might lead to miss gain opportunities.

5.2.3 Continuous local search

Local search is usually used when trying to improve a solution or to find a feasible one in a decision problem. It has been used for the VRP and DVRP as discussed in chapter 2. BFIOQ also uses *local search* on turns trying to improve the initial solution found by BFI. In BFIOQ example, however, the neighborhood for the search has been limited to avoid spending too much time in computation. FIP also uses a *local search* when re-inserting phased-out missions. A *local search* is defined as a neighborhood and the objective is to search for solutions in this neighborhood that might have better quality. In a dynamic problem, *local search* may be used to try to improve the solution. It can be employed continuously during periods when no *routes computation* runs. A first *routes computation heuristics* may be used to find a feasible solution quickly and the Continuous *local search* conducted during the execution of the routes to improve it dynamically. This is for instance the solution presented in [69]. They use a double-horizon strategy with an insertion heuristic that computes a feasible solution for the short term. In the long term, on the second horizon, a Continuous improvement heuristic is run. Applying this approach to our problem would be difficult since the dynamicity is very different. The time between dynamic events is shorter in our case and there is less time to conduct the *local search* before being 'interrupted' by a new dynamic event release. In our problem, the Continuous *local search* could be conducted during time intervals between *routes computations*, which can be a very small period in the types of scenarios we want to cover. The *local search* might not have enough time to find an improvement. The purpose of the short-term horizon is to find a feasible solution quickly. This operation could be handled by the *routes computation heuristic* presented above.

A continuous *local search* heuristic is presented. The principle of the heuristic is to exchange missions of the routes. One tries to find a positive exchange between 2 missions. A positive exchange is characterized by a benefit on the objective score. When testing an exchange, the value of the entire objective function eq. (4.11) with the hypothetical new routes is computed. If this objective value is lower than the current one, the exchange is proceeded. Once an exchange has been processed, the heuristic starts from scratch. Every possible exchange set of size 2 (between 2 nodes) is tested. If none is positive, exchange sets of size 3 are tested. As long as a positive exchange set is not found, the size of the sets is increased until it reaches s_{max}: the total number of missions that can be exchanged considering the RCP. To compute s_{max}, each missions in the routes outside the RCP are counted. This mechanism avoids staying stuck at a local minimum. The Search continues until all exchange sets are tested without finding an improvement in the total objective score. The main constraint we impose is that regrouping missions are allowed but splitting them is not. It could be a lead for further studies.

Hereunder is characterized a valid exchange set of size n without splitting the mission. A valid exchange set is a set where the exchange is possible but not necessarily beneficial. Let's call E^n an exchange set of size n so that $E^n = [\![1, n]\!]$ a set of indexes of missions with d_i the quantity of mission i.

Valid Exchange Set:
$\exists E'^1, \ldots, E'^l$ with $l \in [\![2, n]\!]$ a partition of E^n,

$$\sum_{i \in E'^a} d_i = \sum_{j \in E'^b} d_j, \quad \forall a, b \in [\![1, l]\!]^2 \tag{5.7}$$

For example, the set $E^3 = \{1, 2, 3\}$ with $d_1 = 2$, $d_2 = 1$ and $d_3 = 3$ is a valid exchange set. $E^3 = \{1, 2\} \cup \{3\}$ and $d_1 + d_2 = d_3$

Once the set is validated, the exchange is tested and we check if the hypothetical solution fulfill the deadline constraints. Only if it does, the objective score is computed to evaluate the potential positivity of the exchange.

In case several exchange combinations are valid inside a set, all possibilities are evaluated. If we take the example of a set $E^3 = \{1, 2, 3\}$ with $d_1 = 2, d_2 = 2$ and $d_3 = 2$, there are 2 possible exchange combinations as illustrated in fig. 5.5:

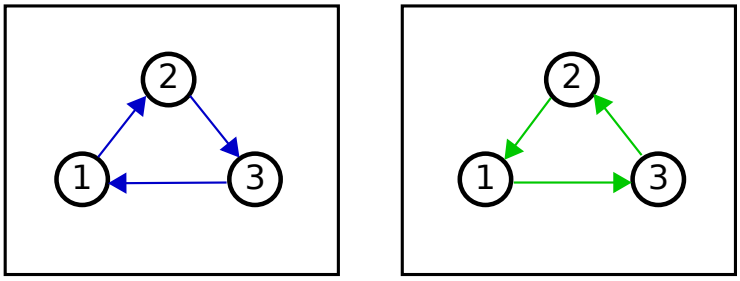


Figure 5.5: Valid exchange combinations example

However, a continuous search such as this one cannot be evaluated in simulated time. They rely on the free time between *routes computation* to try to improve the current routes. This continuous search heuristic has not been evaluated in this work, since the chosen evaluation process chosen using the simulator from section 3.2 works with simulated time. Furthermore, the evaluation process counts on various and numerous dynamic scenarios and runs them in real-time, which would induce very important simulation time.

5.3 Evaluation

The evaluation process for the dynamic problem is operated through simulations. The simulator presented in section 3.2 is used to run dynamic scenarios. A dynamic scenario is built from the graph modeling data about the impacted area and dynamic events. For the simulations, **new demands** and **delays** are considered. The other types of dynamic events have been implemented and their integration into the problem is handled by the solutions but the choice has been made to focus on a subset of these dynamic events to facilitate the analysis process. Real-case data about dynamic events is not available. To create dynamic scenarios, data on dynamic events have to be generated and these two types of events are the most appropriate to generate data that is relevant to the case study of Luchon 2013.
To generate data on dynamic events, metrics are used to control the characteristics of dynamic scenarios.

5.3.1 Metrics for Vehicle Routing Problem

Several metrics are used in the literature to measure dynamism. In the common language, dynamism describes the capability of moving. In the VRP literature it translates with how information variates but the notion is fuzzy which explains why several metrics co-exist to measure dynamism. The most common was introduced by [63], the degree of dynamism (dod) and its variations are presented in the following section.

Degree of Dynamism

This metric measures the proportion of dynamic demands compared to the total number of demands:

$$dod = \frac{number \quad of \quad dynamic \quad demands}{total \quad number \quad of \quad demands} \tag{5.8}$$

The time horizon defines the period T between the beginning of the problem and the end of the planning period. The *dod* does not translate how demands are dispatched on the time horizon, which appeals to another metric. Larsen in its thesis ([63]) also defines the *edod*. It measures an average of how late the demands are released on the time horizon. Let $[\![1, \ldots, n_{imm}]\!]$ be the interval of

indexes of dynamic demands and n_{tot} be the total of demands. The release date is r_i for a demand i.

$$edod = \frac{\sum_{i=1}^{n_{imm}} (\frac{r_i}{T})}{n_{tot}} \tag{5.9}$$

A variation of this metric best suited for VRPTW has also been defined. As a reminder, the set of demands is noted $\mathcal{V}^\star$, the release date for each node $i \in \mathcal{V}^\star$ is noted r_i and the deadline f_i:

$$edod_{TW} = \frac{1}{|\mathcal{V}^\star|} \sum_{i=1}^{|\mathcal{V}^\star|} \left(1 - \frac{f_i - r_i}{T}\right). \tag{5.10}$$

Dynamism

[98] argues that the $edod_{TW}$ is not sufficient to measure dynamism and defines another metric named **dynamism**. For example, fig. 5.6 displays two different dynamic scenarios. Each scenario has two dynamic demands over the time horizon. Demands from both scenarios have the same time windows. Therefore the two scenarios have the same $edod_{TW}$ even if the scenarios show very different dynamics. In the first case, two missions need to be served simultaneously whereas in the second case, the mission is not in competition. Rescue teams can focus on serving the first mission, then focus on the second once it is released.

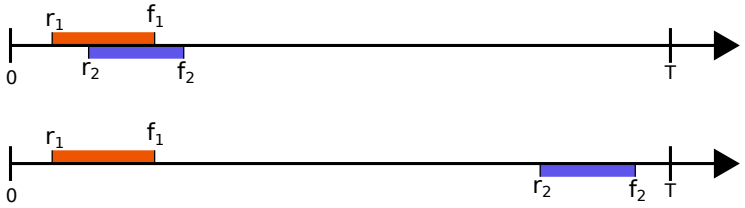


Figure 5.6: Examples of scenarios resulting with the same $edod_{TW}$

It is suggested to use the $edod_{TW}$ to measure **urgency** instead of **dynamism**. Contrary to the $edod_{TW}$, dynamism only considers release dates and not values of the time windows. This metric seems more accurate to our problem since the main challenge is to serve several demands simultaneously and not to serve a demand before its deadline. It means that we prefer to measure the quantity of demands to serve in parallel instead of the time allocated for each service. The authors in [98] offer another metric to measure **dynamism** instead. The presented approach is based on limit cases. A 100% dynamic case is used as a baseline. This case is when the dynamic events are perfectly distributed on the time horizon. On the opposite, the 0% dynamic case is when all demands have the same release date. Both cases are illustrated in fig. 5.7. In this graphic, the second case shows releases that are spaced a little but are actually released on the same date.

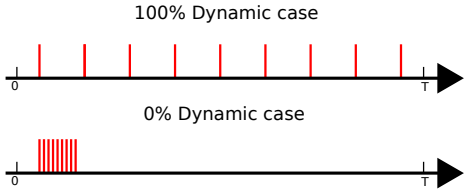


Figure 5.7: Examples of the 100% case and 0% case dynamism

Let the inter-arrival time be the time between two releases. The set Δ of all the inter-arrival dates is defined as:

$$\delta_i = r_j - r_i, \quad j = i - 1, \quad \forall i, j \in \mathcal{V}^\star \tag{5.11}$$

$$\Delta = \{\delta_1, ..., \delta_{|\mathcal{V}^\star|-1}\} \tag{5.12}$$

Hence in the 100% case, for all $i \in \mathcal{V}^\star, \delta_i = \theta$ with $\theta = \frac{T}{|\mathcal{V}^\star|}$ the perfect inter-arrival time as shown in fig. 5.8.

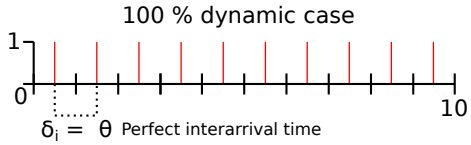


Figure 5.8: Example of the 100% case

Using the value of θ as a reference, a deviation σ_i is defined on the values of the δ_i, for all $i \in \mathcal{V}^\star$:

$$\sigma_i = \begin{cases} \theta - \delta_i & \text{if} \quad i = 1 \quad \text{and} \quad \delta_i < \theta \\ \theta - \delta_i + \frac{\theta - \delta_i}{\theta} \cdot \sigma_{i-1} & \text{if} \quad i > 1 \quad \text{and} \quad \delta_i < \theta \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

As one can see, this deviation σ_i is defined in regard to the previous release. Then the first release that does not have a previous release to refer to, is defined only according to its inter-arrival time with the next release. This is the first line of eq. (5.13). In the second line is defined the deviation in all other cases except when the inter-arrival time exceeds the perfect inter-arrival time θ. In this case, deviation is set to 0 and the next deviation is impacted. The total deviation on the entire scenario is defined as:

$$\tau = \sum_{i=1}^{|\mathcal{V}^\star|} \sigma_i \tag{5.14}$$

In order to define the dynamism, values of deviation are normalized by the maximum deviation:

$$\tau_M = \sum_{i=1}^{|\mathcal{V}^\star|} \overline{\sigma_i}, \text{ where} \tag{5.15}$$

$$\overline{\sigma_i} = \theta + \begin{cases} \frac{\theta - \delta_i}{\theta} \cdot \sigma_{i-1} & \text{if} \quad i > 1 \quad \text{and} \quad \delta_i < \theta \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

Finally the **dynamism** is:

$$Dynamism = 1 - \frac{\tau}{\tau_M} = 1 - \frac{\sum_{i=1}^{|\mathcal{V}^\star|} \sigma_i}{\sum_{i=1}^{|\mathcal{V}^\star|} \overline{\sigma_i}} \tag{5.17}$$

Another metric, built on the same principle, is defined in order to measure the spatial distribution of the total number of victims over the nodes of the problem.

Quantity Distribution Index

This metric is called Quantity Distribution Index (QDI) and is also based on boundary conditions (0% and 100% case). Here it is a 100% distributed case where all the nodes of the problem would host the same number of victims. In opposition, the 0% distributed case is when every node hosts one victim but one node hosts the rest of the total of victims. In this work, the total number of victims is a function of the size of the problem (number of nodes). In fact, in order to evaluate the solutions to scenarios that are relevant to the case study of Luchon 2013, the total quantity has been set proportionally to the total number of victims in Luchon. An example is illustrated in fig. 5.9, where the total number of victims is 12, dispatched on 3 nodes. The 100% distributed case is when each node hosts 4 victims whereas, in the 0% distributed case, there would be 10 victims at one node and 1 at the others.

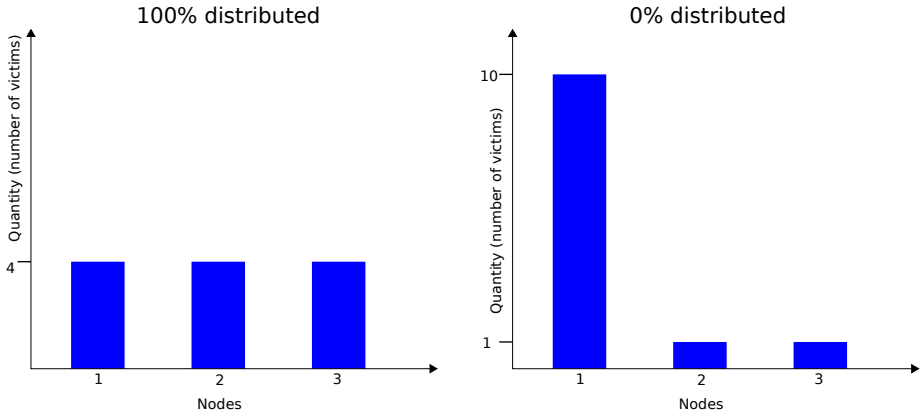


Figure 5.9: Examples of the QDI limit cases

However, contrary to the release dates, the number of victims are integer values, hence the 100% case is not reachable in all scenarios – it depends on the total number of victims and its divisibility by the number of nodes. β is the perfect quantity. It is the quantity of all demands in the 100% distributed case:

$$\forall i \in \mathcal{V}^\star \quad \beta = \frac{\sum_{i=1}^{|\mathcal{V}^\star|} q_i}{|\mathcal{V}^\star|} \tag{5.18}$$

A deviation from the 100% case is defined through ρ_i's, where $i \in \mathcal{V}^\star$. It is the difference between the quantity at node i and β:

$$\forall i \in \mathcal{V}^\star, \quad \rho_i = |\beta - q_i| \tag{5.19}$$

The maximum deviation is used to normalize the deviations. It is based on the minimum quantity of 1. The maximum quantity is reached when all nodes but one have their quantity at 1.

So, the maximum deviation is:

$$\overline{\rho_i} = \begin{cases} \beta - 1 & \text{if} \quad q_i < \beta \\ (|\mathcal{V}^\star| - 1) \cdot (\beta - 1) & \text{otherwise} \end{cases} \tag{5.20}$$

In fact, if we consider the deviation in the 0% distributed case, every demand has a quantity of 1 which is a deviation of $\beta - 1$ from β and one demand gathers the rest of the quantity. This quantity equals the sum of the deviation $(\beta - 1)$ for all other demands $(|\mathcal{V}^\star| - 1)$ at which we add the quantity of the demand itself (β). So the deviation is $(|\mathcal{V}^\star| - 1) \cdot (\beta - 1)$.

And finally, the QDI is:

$$QDI = 1 - \frac{\sum_{i=1}^{|\mathcal{V}^\star|} \rho_i}{\sum_{i=1}^{|\mathcal{V}^\star|} \overline{\rho_i}} \tag{5.21}$$

These metrics are used in this work in order to control the characteristics of the scenarios generated. The following section presents how the dynamic scenarios are generated for the experimental set.

5.3.2 Dynamic scenarios

As mentioned earlier, the dynamic part of a scenario is composed of the graph and dynamic events. First, to generate the graph structure, the generator, presented in section 3.1, is used. We generate a graph with 3 zones with the following characteristics:

These parameters have been chosen to model the characteristics of Luchon territory. For each node, the generated inputs follow the same process as the static study:

- **Priority**: Generated using an uniform distribution among the 4 priority values that are used in practice by rescue teams. The values of the deadlines are set accordingly, with the priority of the nodes as follows:

Zone	1	2	3
Center	0,0	0,0	0,0
Size (in meters)	1000	2000	4000
Density (nodes per km^2)	10	5	1
Degree	3	2	1

Table 5.4: Graph generation parameters

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

Table 5.5: Variables

- **Service time**: Uniformly distributed between 300 and 2100 seconds. This range has been determined after discussion with rescue teams from SDIS 31.
- **Category**: The proportion of nodes for each category has been extracted from Luchon 2013's Experience Feedback EF. The experimental sets follow this proportion.

Sizes of demands are generated using a different approach. To be able to control the QDI of the set of demands, numerous data series are generated. It is difficult to randomly generate a data set with a given QDI and Dynamism. Hence data sets for release dates and quantity distributions are generated and sorted by *dynamism* and QDI respectively. Then data series are picked in the right space for the desired values of the metrics. The goal is to have, for every value of QDI, a set of data series to extract from when one wants a quantity distribution series for a given QDI value. To generate quantity distribution series that cover the full spectrum of values, different generation laws are used. For each of these laws, 10^4 series of this law are drawn. Let us note P the total number of victims.

- For low QDI, a truncated normal law is used. The mean equals $a \cdot \frac{P}{2}$ and the standard deviation equals $b \cdot \frac{P}{4}$. The best values for a and b to cover the range of quantity distribution are $a = 1$ and $b = 1.6$. These values have been found by experiment.
- For high QDI, the distribution is based on a truncated normal law whose values are selected to fit this range of distribution.
- For medium QDI, a truncated normal law of mean $a \cdot \frac{P}{\mathcal{V}^\star}$ and standard deviation of $b \cdot P$ is used.

Once the data series are generated, they are sorted by QDI and stored in a database.

The same process is executed for the generation of time series associated with a given dynamism. The time series correspond to the release dates for the dynamic demands r_i. The time horizon T has been set to 4 hours but the series could be scaled up or down. 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 each parameter has been tuned to generate time series in the desired dynamism range. The following parameter values have been set. The amplitude a is set to 1.05, the frequency f is defined as $2.5 \cdot 10^{-4}$ and 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 $1.2 \cdot \frac{T}{V}$. This law has been used by [98].

Figure 5.10 displays the database that we built for release dates time series. The graphic shows the number of data series for each range of size 5 of dynamism. The different generation laws are displayed. The laws are coupled to cover the full spectrum of dynamism values.

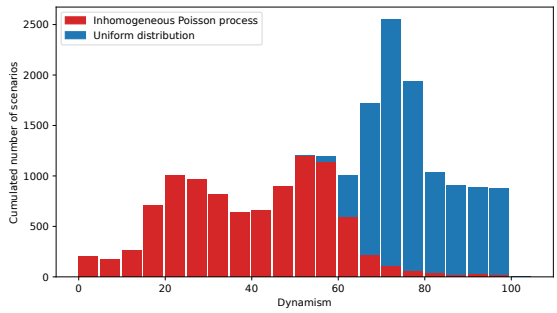


Figure 5.10: Dispatching of time series by dynamism according to the generation process

When one selects a data series for a given dynamism or QDI, it is picked in an interval of 5 width around the desired value. The data of the graph and the dynamic demands have been generated but as mentioned above, we also study **delays** in experimentation.

delays are signaled by vehicles during the crisis and the vehicle's process in the simulator. The rate of delays of 15% among all interventions has been communicated by rescue teams. To make the delays reproducible and fair to

every solution, one cannot generate a delay online at every arrival or departure from a node with a chance of 15%. Hence the time horizon has been divided into a range of 10 minutes. A table of delays is created with a delay value for every 10 minutes. It means that for a given demand, the delay value (null or not) is the same for every period of 10 minutes. When the value is non-null, the range for the delay in travel times is between 60 and 900 seconds with a uniform distribution. For service time delays, the value is between 10 and 120 seconds and this value is multiplied by the number of rescued victims during this operation. In the simulator, when a vehicle arrives at a node, it reads the value of the travel time delay in the table corresponding to the planned date of arrival at the node. The same process is executed for service time delays when a vehicle leaves a node.

5.3.3 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 computation time for BFIOQ, that is theoretically the more complex *routes computation heuristic* , overall computations (more than 50 per scenario) and all the scenarios has been measured under 4 seconds. To size this value to the problem's scale, the quadratic complexity of BFIOQ is used. For example for a problem of size 180 nodes, the initial RCP is calculated as follows:

$$RCP(180) = RCP(60) \times \left(\frac{\mathcal{V}^\star}{60}\right)^2 = 4 \times \left(\frac{180}{60}\right)^2 = 36 \; seconds \qquad (5.22)$$

The advantage of setting the initial RCP from experiments measures and scaling is to limit systematic overtimes at the beginning of the simulation in case the value is too low. On the contrary, if the value is too high, it could waste time to compute solution on a too-large period.

The process that generates experimental sets is independent of the size of the problem. For experimentation, several sets of different scales are built.

Note that in practice, **delays** are also used when everything rolls according to the plan. At every arrival or departure from a node, **delays** with null values are emitted. It allows updating the state of the vehicles. They are also logged in the output file.

Luchon scale experiment

The first set is a Luchon scale set with a total of 60 nodes. For this set the following vehicles have been considered:

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. It makes a total of 360 scenarios in the set. The 3 presented *routes computation heuristics* are evaluated under the 3 introduced

Category	1	2	3	4	5
Number of vehicles	5	4	4	1	1
Vehicles capacity	30	10	5	1	10

Table 5.6: Vehicles characteristics Luchon scale

routes computation strategies for a total of 9 combinations. The scenarios from this set are run using the simulator. It is 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 sciences ([11]). Experiments with 6 containers in parallel have been driven on nodes with 2 CPUs Intel Xeon E5-2660 v3 of 8 cores and 128 GiB of memory. For all the experimental results, if the tested solution does not find a feasible solution at any moment of a simulation, it is recorded as an error. For the Luchon scale experiment, no error has been logged.

Figure 5.11 displays the evolution of the objective score according to the dynamism for all 9 combinations of *routes computation heuristics* and *routes computation strategies*. The lines between points in this figure and further ones in this chapter are not used to approximate the behavior of solutions between measures, but only to ease the reading of the data displayed if one wants to follow a specific combination of *routes computation heuristic* and Strategy.

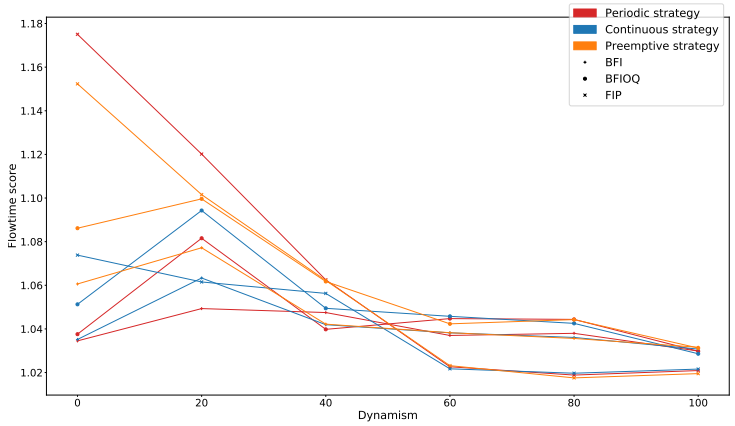


Figure 5.11: Objective score according to dynamism

As specified in the legend, the *routes computation heuristics* are represented with different colors and the *routes computation strategy* with different markers. In this figure, the objective score, *i.e.* the quality of the solutions, is displayed. The objective score is the value of the objective function in which we sum all the demands that are served during the simulation. For this figure, the mean of the objective score is computed on all the scenarios of the given dynamism, what-

ever 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 have been **normalized by the best *routes computation heuristic* /*routes computation strategy* scenario by scenario**. The graphic is difficult to read but a first observation is made. At high dynamism (over 60) the curves are more regrouped by *routes computation heuristics* and the differences are smaller than at low dynamism. Furthermore, it seems that BFI and BFIOQ show significantly and comparatively better results at low dynamism whereas FIP quality is only slightly better at high dynamism. Figure 5.12 displays the same results but the *routes computation strategies* with each *routes computation heuristics* have been merged into one curve. It allows observation of only *routes computation heuristics* results. The observations are confirmed with this graphic.

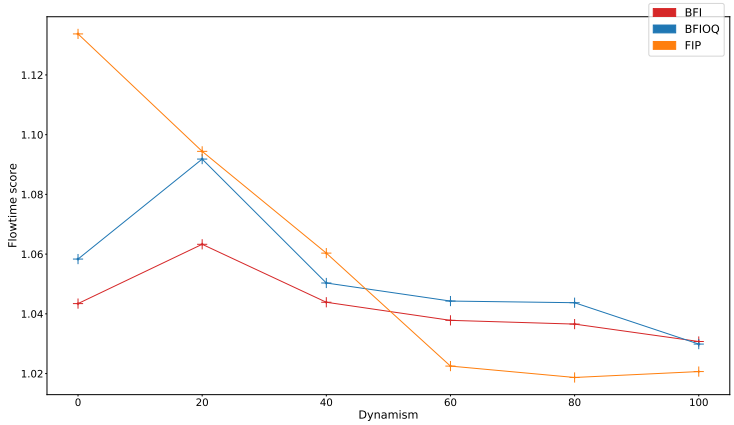


Figure 5.12: Objective score according to dynamism merged by *routes computation heuristic*

At low dynamism, the demands are released by batch in a short amount of time. In these situations, the re-optimization approach has more opportunities for gain when building back the routes from scratch. It sustains the hypothesis that BFI and BFIOQ have better static performances. Scenarios with the dynamism of 0 are almost static problems. For high dynamism values, FIP shows slightly better performances. When the demands are more distributed over the time horizon, there are fewer opportunities for gain since a part of the routes is already being served. On the opposite, the time between two new demands is steadier. The best computation time performances of FIP that are observed in fig. 5.13, become a greater advantage, higher computation time may lead to make demands wait longer in the buffer and maybe miss an opportunity to be inserted at a good position in the routes.

This graphic may also explain the better performances of BFI compared to BFIOQ but the differences are really small. These arguments are confirmed by fig. 5.14 that displays the final RCP value according to dynamism. Higher

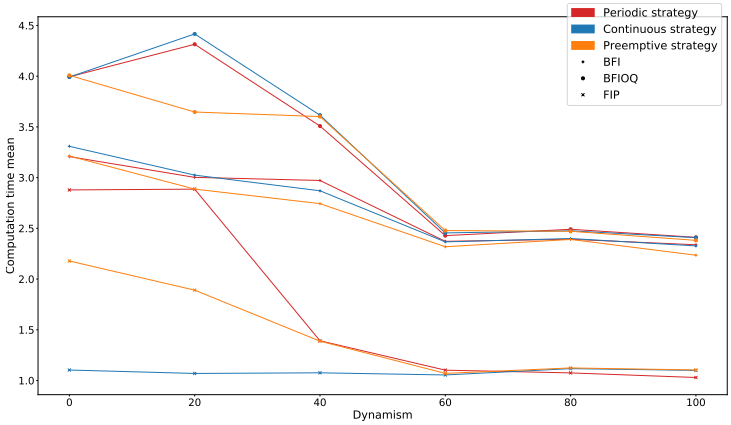


Figure 5.13: Computation time according to dynamism

computation time leads to more overtime and therefore a bigger RCP. As a reminder, every time the RCP is violated through an overtime, the RCP value is doubled. If the RCP is higher, more missions are fixed, which lets fewer opportunities to improve the routes.

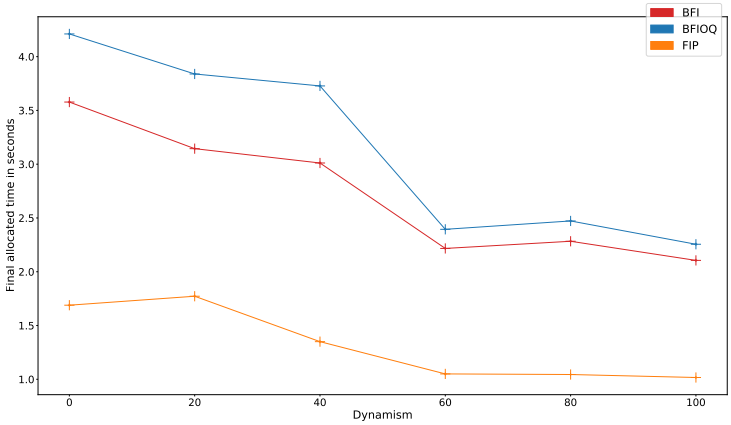


Figure 5.14: Final value of the RCP according to dynamism

Figure 5.15 displays the same results as fig. 5.11 but merged by *routes computation strategy*.

At high dynamism, the three *routes computation strategies* are equivalent but at low dynamism, the **Continuous** *routes computation strategy* displays better quality than the **Periodic** *routes computation strategy* which is better than the **Preemptive**. When the demands arrive by batch, the Continuous *routes computation strategy* updates the routes more often whereas the Periodic

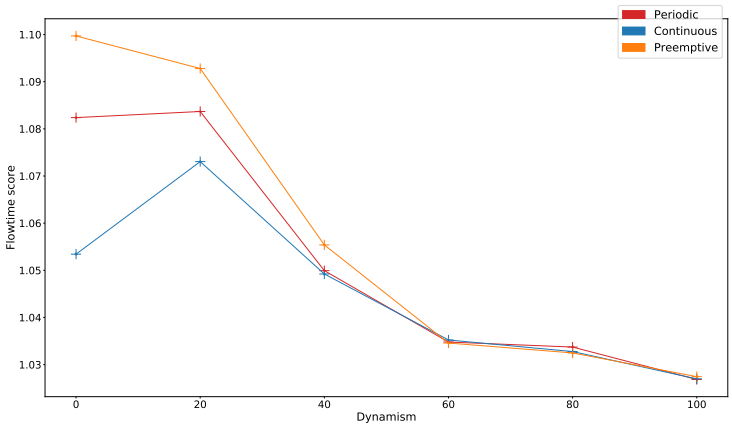


Figure 5.15: Objective score according to dynamism merged by *routes computation strategy*

routes computation strategy waits for the end of the RCP to re-compute and may miss gain opportunities. The Preemptive *routes computation strategy* is bad because the demands arrive very regularly in a short period which may lead to many *routes computation* interruptions and no update of the routes for a long period.

The influence of Dynamism has been observed but the impact of QDI is also evaluated. In fig. 5.16, the QDI seems to have opposite impact than dynamism.

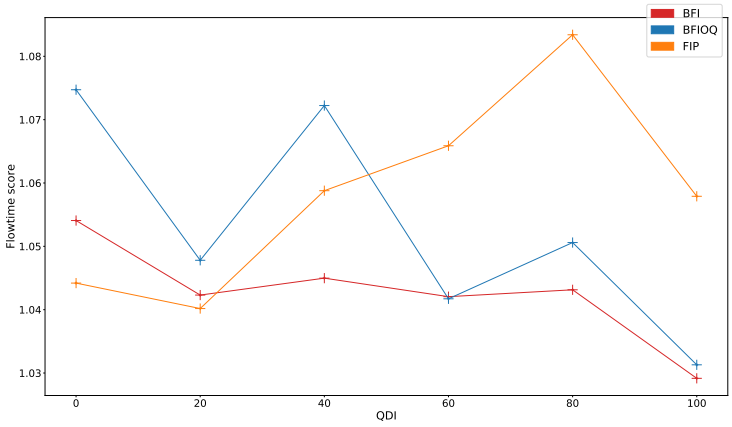


Figure 5.16: Objective score according to QDI merged by *routes computation heuristic*

For low distribution scenarios, FIP shows better performances than *routes computation heuristics* of the re-optimization approach. At low QDI, most de-

mands are small but few demands are big. FIP is better to insert a big demand. For example, if the best position for this demand is at the end of the routes, FIP only has to test all the positions once to get the best way to insert it and directly insert all at the end. On the opposite, if BFI and BFIOQ try to insert one big demand, every position will be tested and at each time the demand is segmented. Furthermore, all the other small demands represent very small opportunities for gain. This advantage is small and reversed at higher distributions. The re-optimization of *routes computation heuristics* is better with more balance demands since there are better opportunities for optimization in insertion.

The distribution does not seem to have a clear impact on the *routes computation strategies* quality.

The observations at the Luchon scale and the compared qualities of solutions might vary with the size of the problem.

Scale-up experiment

Another experiment has been built to observe the impact of resources degradation. When resources are too short to serve all demands in one trip because of capacity limitations, the state of the problem is qualified as degraded.

The size of the problem has been increased without increasing resources. By increasing the scale without scaling up the resources, the situation is more difficult to handle for rescue teams. This experimental set is built only on two values of dynamism (20, 80) and one value of QDI (60). These values have been chosen because the results of the Luchon scale experiments displayed clear differences for these values and fewer variations. For example, fig. 5.12 shows the evolution of the objective score according to the dynamism for a QDI of 60. There is a clear variation where FIP is worst at a dynamism of 20 and becomes better than the re-optimization *routes computation heuristic* at 80.

The results, merged by *routes computation strategies*, do not show any clear advantage for a *routes computation strategy* in terms of resilience to errors as observed in fig. 5.17.

In this figure, every point is the number of scenarios that ended with an error over the 30 scenarios, tested for each size of the problem. We clearly observe as expected that the number of errors increases but no *routes computation strategy* seems to better handle the errors. Figure 5.18 displays clearer results. FIP *routes computation heuristic* is less resilient to errors (mainly deadline violations) than the re-optimization *routes computation heuristics*. At sizes 100 and 120, it makes more than twice more errors. In fact BFI and BFIOQ re-compute from scratch at every step and are less impacted by bad decisions than FIP. On the opposite, FIP, does not question previous decisions that could have a bad impact on feasibility, if this decision has a good score.

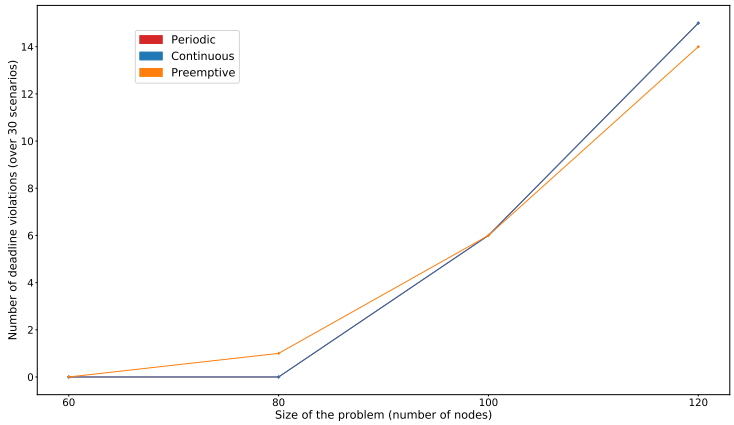


Figure 5.17: Number of errors over 30 scenarios QDI=60 / dynamism=80 merged by *routes computation strategy*

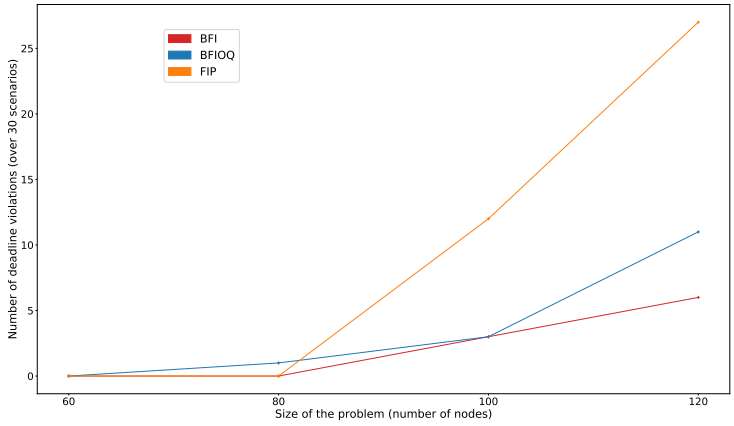


Figure 5.18: Number of errors over 30 scenarios QDI=60 / dynamism=80 merged by *routes computation heuristic*

Large scale experiment

A Large scale experiment has been conducted and contrary to the scale-up experiment, resources have been scaled up from available resources in the Luchon scale experiment. The Large scale is 3 times higher than the Luchon scale. In order to avoid facing a complex situation, the resources have been multiplied by 3 as well. The same generation process has been followed to build this experimental set. However, the time Horizon is the same (4 hours).

Contrary to the Luchon scale experiment, many errors are recorded during the simulations. As mentioned above if a scenario leads to an error with any of

the solutions (*routes computation heuristic* + *routes computation strategy*), it is removed from the set of displayed results. If for a certain value of the evaluated metric, all scenarios drive at least one solution to error, the point is not included in the graphic. This is the case in fig. 5.19 where the scenarios of dynamism 20 are not displayed. A first observation is made. The **Preemptive** *routes computation strategy* shows very bad performances at low dynamism. Furthermore, it seems to be the worst *routes computation strategy* for any dynamism value. In this figure, there is no data for the dynamism of 20 since, for every scenario of this dynamism value, at least one solution produced an error.

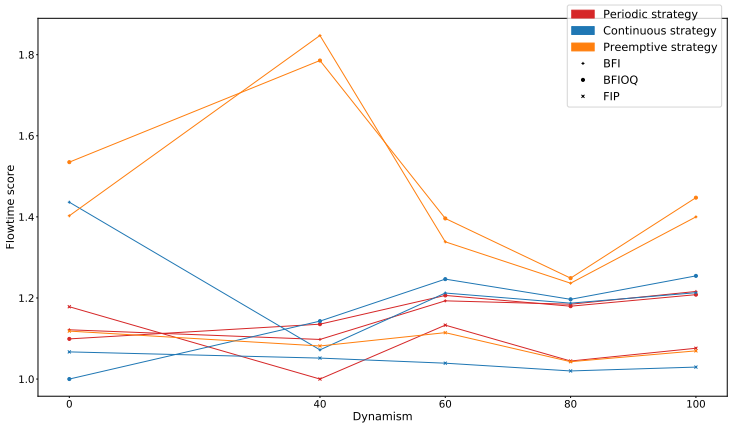


Figure 5.19: Objective score according to dynamism

Figure 5.20 displays the rate of error according to distribution with a fixed value of dynamism of 20 which is where the problem is observed in fig. 5.19. Scenarios are merged by *routes computation heuristics*. Each point represents 30 scenarios.

It shows that the error rate is almost always at 1 for FIP *routes computation heuristic* . For the distribution values where FIP error rate is not at 1, the error rate of the other *routes computation heuristic* complements one which justifies the lack of reliable scenarios to display in fig. 5.19. Figure 5.21 zooms out from fig. 5.20 in order to display the evolution of the error rate according to dynamism.

It shows that FIP is really bad in resilience to errors at low dynamism. Also, the dynamism has a high impact on error rate for this *routes computation heuristic* because, on the opposite, the error rate is really low at high dynamism. These results confirm the observation of the Luchon scale experiment in fig. 5.12. FIP shows bad performances in terms of solution quality at low dynamism. Figure 5.21 displays correlated results on error rate. These observations are related because at some point if the *routes computation heuristic* returns solutions with too bad qualities, it leads to bad decisions. Past decisions being less questioned by FIP than with re-optimization *routes computation heuristics*, also leads to

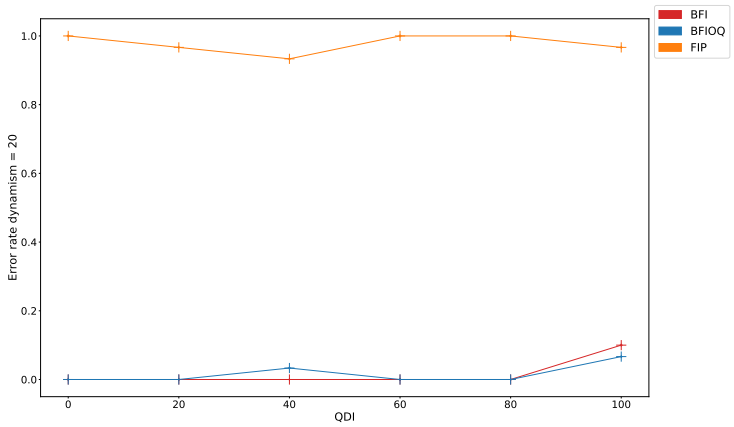


Figure 5.20: Error rate according to QDI dynamism = 20 merged by *routes computation heuristic*

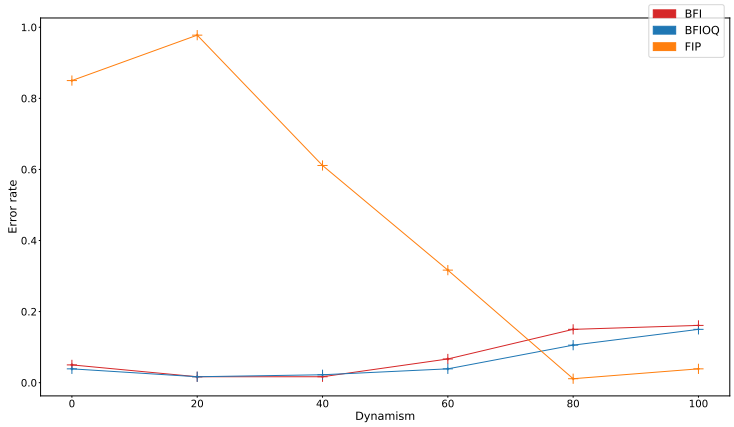


Figure 5.21: Error rate according to dynamism merged by *routes computation heuristic*

deadline violations. As a reminder, at low dynamism, numerous demands have to be inserted into the routes in a short amount of time, which, coupled with the *routes computation strategies* is almost equivalent to having several static problems in a row. FIP is an insertion heuristic for dynamic problems and shows bad results at low dynamism. At high dynamism values FIP shows however very good resilience to error with almost a rate of 0. It is even better than BFI and BFIOQ. It may be explained by the computation time performance that is better for FIP at high dynamism. It might be observed in fig. 5.22 with an example with all the solutions with a focus on dynamism = 80 or in fig. 5.23 which also shows the computation time, merged by *routes computation heuristic*

, according to Dynamism.

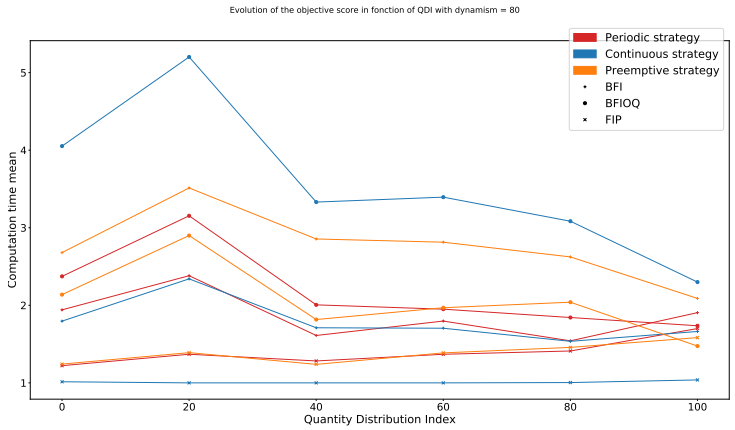


Figure 5.22: Computation time according to QDI dynamism = 80

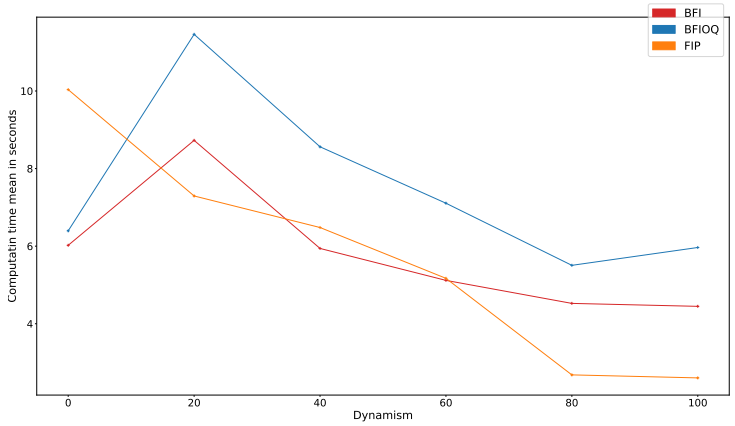


Figure 5.23: Computation time according to dynamism merged by *routes computation heuristic*

The re-optimization *routes computation heuristics* however are not impacted this much by the evolution of dynamism. A slight increase in the error rate may be observed at high dynamism but this is not explained by the computation time. The evaluation reveals a high impact of dynamism on solution resilience (capacity to find feasible solutions and avoid errors). Figure 5.24 exhibits that the QDI does not affect the resilience, except for QDI = 100 for the re-optimization *routes computation heuristics*.

For this limit case, when inserting a big demand, we might have to report several small demands later in the routes or plan this big demand at the end

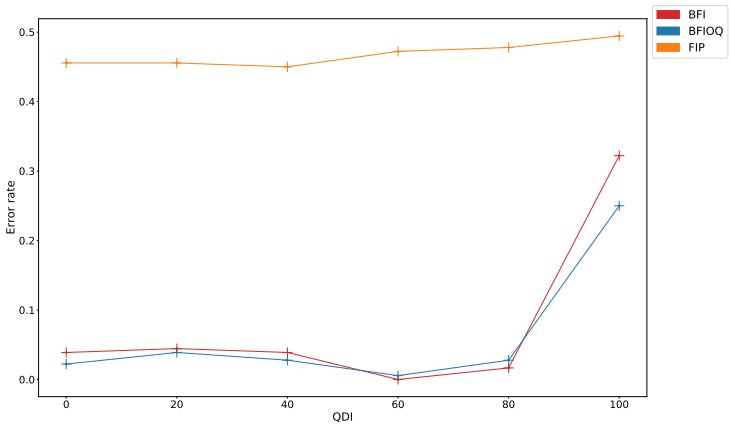


Figure 5.24: Error rate according to QDI merged by *routes computation heuristic*

of the routes which could lead to deadline violations. In many cases, the error rate for the re-optimization *routes computation heuristics* is always lower than FIP's by far.

The analysis of errors in fig. 5.25 confirms the results of the Luchon scale concerning the *routes computation strategies*. **Continuous** *routes computation strategy* is more resilient to errors. The resilience of the **Periodic** and **Preemptive** *routes computation strategies* are very close.

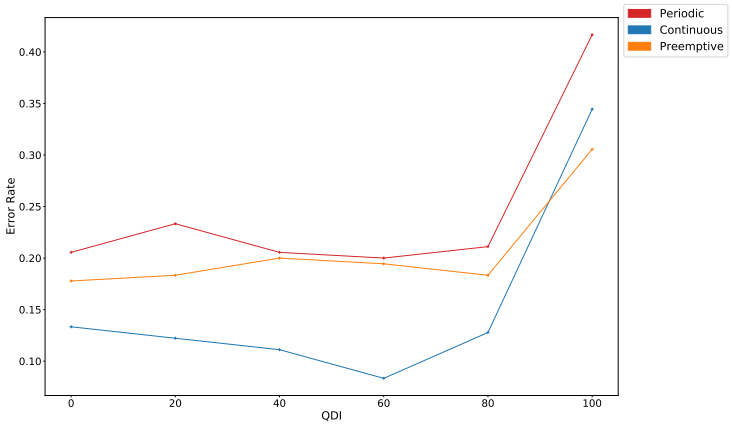


Figure 5.25: Error rate according to QDI merged by *routes computation strategy*

Based on the knowledge of the results of errors, the analysis of objective scores is conducted cautiously. Figure 5.26 presents the evolution of objective

score according to QDI.

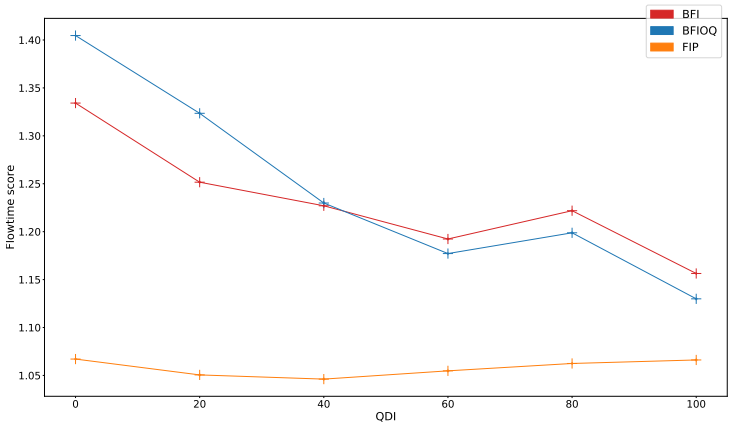


Figure 5.26: Objective score according to QDI merged by *routes computation heuristics*

These results (fig. 5.26) exhibit better solutions quality for FIP. It is confirmed at high dynamism by fig. 5.27 which is a zoom on the scenarios with a Dynamism of 100.

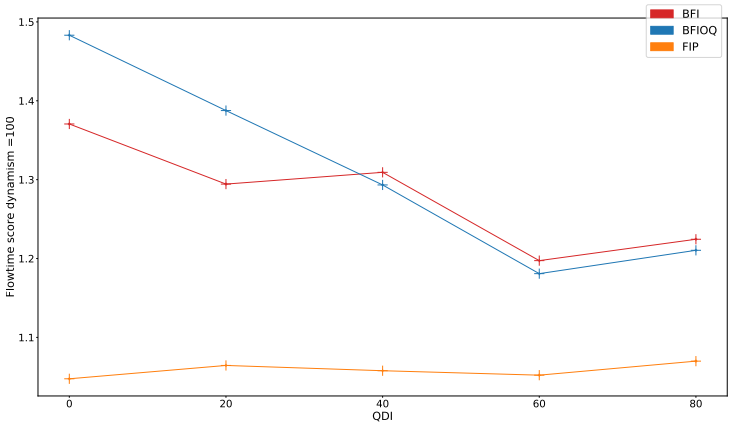


Figure 5.27: Objective score according to QDI merged by *routes computation heuristics* dynamism = 100

Hence at Large Scale FIP shows better performances in terms of solution quality. However, it is not resilient to errors and is not reliable at low dynamism.

5.4 Conclusion

In this chapter, the characteristics of the dynamic formulation of the problem have been introduced. The different dynamic events of the model have been presented and the mechanisms to handle the dynamic *routes computation* have been detailed.

- 3 strategies for *routes computation*: **Continuous**, **Periodic** and **Preemptive** *routes computation strategies* have been defined.
- 2 different approaches have been studied. A re-optimization approach that adapts BFI and BFIOQ, initially developed for the static version of the problem has been proposed. FIP, an insertion heuristic that inserts demands in the existing routes has been introduced.

Using the simulator from section 3.2 and the graph generator from section 3.1, several experiments have been conducted:

- Luchon scale data set: An experimental set built to model similar characteristics to the case study of Luchon.
- Large scale experiment: A set of dynamic scenarios three times bigger in several nodes and victims than the Luchon crisis. For this experiment, the resources have been increased proportionally to the scale increase.
- Scale-up experiment: An experimental set built by increasing the scale of the problem, but with constant resources.

The analysis of the results from these experiments shows that BFI and BFIOQ, used in a re-optimization approach, offer better solution quality at the Luchon scale. These *routes computation heuristics* also proved to be more resilient to errors (when no feasible solution is found) as experienced in a Scale-up and at a Large Scale. FIP however displayed better solution qualities at high dynamism but is not reliable at low or medium values of dynamism because of its error rate. The experiments also showed that the Continuous *routes computation strategy* gives better performances. These results motivate us to consider a hybrid approach that could be tested in further work. The used solution could be chosen according to the estimated size and dynamism of the problem. Furthermore, the fact that the Continuous *routes computation strategy* is the best strategy among the tested *routes computation strategies* motivates to consider a Continuous Search approach. The *local search* that has been presented but could not be evaluated properly could be used to try to improve the solution during the period of the crisis when no *routes computation* is in progress. The main difficulty that was faced to evaluate this option is that simulating an experiment with this type of approach can not be done in simulated time. The simulation process skips the periods when nothing appends (no *routes computation* is in progress) to accelerate simulations. However, the Continuous Search should be used this time to try to improve the current routes, hence evaluation of the Continuous search must be conducted in real-time.

Chapter 6

Conclusion

In this thesis, a crisis management problem is studied. This problem is the optimization of victims' rescue operations in case of flash floods. The objective is to build rescue vehicle routes to locations where victims need to be saved and drive them back to safety. This problem is part of the class of Vehicle Routing Problem (VRP). The contributions of this thesis answer two different formulations of the problem: Static and dynamic.

6.1 Static contributions

In chapter 4, different contributions are presented. These contributions answer to the static formulation of the problem. In this problem, the information is considered to be known entirely at the beginning of operations. In this chapter:

- A Mixed Integer Linear Programming (MILP) model of the problem is presented. The VRP studied in the thesis is developed and the inputs and variables of the problem are introduced. An objective function based on flow-time and prioritization of demands is introduced.

- Heuristics are developed to tackle the problem. Greedy algorithms and more complex heuristics that question previous decisions are introduced. Among them Best Flow-time Insertion (BFI) insert demands in the route of the vehicle – and at the position in this route – where the impact on the global objective score is the lowest. Best Flow-time Insertion with Order Questioning (BFIOQ) is a variation of BFI in which a *local search* on the current turn is conducted after the insertion of a demand in a route.

- Heuristics to optimize resources distribution between different intervention sectors are studied.

- Various evaluations are conducted:

  - Evaluation of the exact method. A branch-and-bound method is evaluated with different scales of problems. This evaluation allows observation of the limit of problems sizes upon which exact methods are not consistent with the computation time constraints of the problem.
  - Comparison of heuristics. Insertion heuristics BFI and BFIOQ display better performances both in terms of computation time and solutions qualities.
  - Resources distribution evaluation. An experiment is conducted to evaluate the different heuristics presented.

The Static formulation is a simplification of the real-life problem. However, it can be useful for training or preparation purposes.

6.2 Dynamic contributions

In chapter 5, the dynamic version of the VRP is studied. In this problem, different dynamic events are considered that are released during the operations. It is closer to the real-life problem. The following contributions have been made in this chapter:

- Heuristics developed for the static problem have been adapted with a re-optimization approach.
- A dynamic insertion Flow-time Insertion Phase-out (FIP) heuristic has been developed. It inserts demands at the spot of plan missions in the route. These missions are then phase-out and re-inserted following the same process.
- A metric from the literature has been presented to measure the dynamism of evaluation scenarios.
- A metric to measure the quantity distribution has been introduced.
- A generation process has been offered to control the values of the two metrics on dynamic scenarios. It has allowed assessing the influence of these metrics on solutions performances.
- Three dynamic *routes computation strategies* to determine when to update the solutions have been proposed.
- Experiments on two scales of problems (60 and 180 nodes), 6 values for each metric, 10 scenarios for each combination of metrics, and with all 9 combinations of the three *routes computation strategies* and 3 *routes computation heuristics* (BFI, BFIOQ and FIP) have been conducted.

  It reveals that the best strategy is the Continuous *routes computation strategy*. The best *routes computation heuristic* is BFI but FIP performs

slightly better at high dynamism values. It raises the question of the performances with higher scales of problem. This question should be answered in further works.

- A set of dynamic scenarios for evaluation. This set contains scenarios with two scales of problems and six values of Dynamism and Quantity Distribution Index (QDI). It can be used in other works as a crisis management benchmark for VRPs.
- A continuous search heuristic has been developed but has not been evaluated.

6.3 Evaluation contributions

In order to conduct evaluations of the solutions developed in the thesis, softwares have been developed:

- A parameterized graph generator to create evaluation instances with specific characteristics.
- A simulator to run dynamic crisis scenarios in simulated time. This software uses a multi-process environment with each actor of the crisis management (vehicles, decision center, and call center) handled by a process. The communication between these actors is also simulated to signal events and send updated routes.

6.4 Perspectives

The contributions presented in this thesis raise different research questions that could be studied in future works.

6.4.1 Other approaches from the literature

In this thesis, the choice has been made to follow a re-optimization approach with static heuristics. This choice has been mainly motivated by computation time. However other approaches could also be tested. Meta-heuristics are generally slower to compute for a static problem, however, in a dynamic environment, the computation time could be relevant to our computation time constraints if the frequency of dynamic events is not too high. For example, the Tabu search presented in [44] gives better results with a higher computation time but the insertion of a single node and the search for an improved solution takes around 7 seconds for instances of size 60. It would be interesting to adapt such an approach to the problem of this thesis.

An evaluation could be conducted on the same scenarios that have been used in the campaign of experiments of chapter 5. It may reveal better performances with certain values of scale, dynamism, and quantity distribution.

A double horizon approach is also interesting to adapt to the problem. Heuristics that compute a feasible solution in a short amount of time have been developed in this thesis and could be used for the short-term horizon. An improvement meta-heuristic could be used for the improvement of solutions on the long-term horizon. In Flow-time Insertion Score (FIS) developed in this thesis, new demands can be inserted at any spot in the routes except inside the Routes Computation Period (RCP). It is not considered a double-horizon approach since there is no clear limitation between two distinctive horizons. The Continuous Search presented in chapter 5 could also be adapted to this approach. The main challenge of this approach would be to define the horizons. The dynamic of the crisis is important. For the Luchon scale, there is a new demand released every 5 minutes approximately on average. It is even less on a bigger scale. The problem is to find the right balance between short-term and long-term to avoid canceling improvement computations too often. A strategy could be defined to decide whether a demand should be inserted in the short-term routes, or if it is not urgent and can be inserted in the long-term horizon routes.

6.4.2 Prioritization

The problem resolution and performances evaluation rely strongly on the prioritization of demands. This prioritization uses a scale of 4 values in the current rescue teams system. A larger scale of priority with more values could help to improve the victim relief operations. The major challenge of this perspective is to be able to put numbers on the different decision criteria of rescue teams from the SDIS 31.

An internship has been conducted on the project to study such prioritization. A post-doctoral researcher is currently studying the benefit of the web semantic approach to prioritization. The approach is based on a knowledge graph that gathers all the data of the crisis and relies on a shared vocabulary represented in an ontology. Then, the reasoning is done with inference rules to provide new knowledge and compute priorities.

6.4.3 Territory resilience evaluation

This thesis has been conducted on the short-term phase of the i-Nondations project which is the Response phase of the crisis management. The long-term phase of the project focuses on the resilience of territories to flash floods and the *build back better*. When a territory is impacted by a flood, its priority is to recover and build back damaged infrastructures. The budget allocated to stricken areas is often not sufficient to include all territory development recommended to improve resilience before the next crisis. If a choice has to be made between different developments, the evaluation of the infrastructures improvements is important to base a decision on.

From that perspective, the dynamic simulation process presented in this thesis can be helpful. Simulations can be run with the different territory devel-

opments recommended to measure which one allows the best improvement for rescue teams' relief operations. An improvement could avoid a road being cut and ease the rescue process. Others might enable to reduce the impact of the flood on some households and therefore change the priority of these demands (since victims can remain longer on the location without being endangered).

The main challenge is to compute the impact of a territory's development on the flood. A module needs to be developed to deduce from a territory development, the modifications to the crisis scenario.

6.4.4 Big data support

As presented in chapter 2, studies have been conducted using big data with heterogeneous data sources to improve data collection for crisis management. This is the case of [36] and [14] for instance. The use of such approaches could transform the studied problem. Data collection can bring more knowledge earlier in the problem. Furthermore, probabilistic information about dynamic demands might be collected, and stochastic approaches could be studied. Prediction approaches could be studied and adapted to the problem as well. For instance, [50] studies the prediction of firefighter interventions. These approaches might help the response phase with anticipation of big quantity demands or high priority demands.

6.5 Publications

Capacitated Vehicle Routing Problem under Deadlines
Florent Dubois, Patricia Stolf and Paul Renaud-Goud. *International Conference on Information and Communication Technologies for Disaster Management (ICTDM)*
March 2020 - Published

Towards Flash Floods Management Using Temporal Loops
Patricia Stolf, Julian Berthet, Christelle Bosc, Pauline Bremond, Arnaud Ceyte, Victor Champonnois, Anne-Laure Collard, Florent Dubois, Georges Da Costa, Katrin Erdlenbruch, Didier Felts, Rémy Gasset, Frédéric Grelot, Christophe Héral, Benjamin Piccinini, Jean-Marc Pierson and Amal Sayah. *Research Report IRIT - Institut de Recherche en Informatique de Toulouse*
June 2021 - Published

A Re-optimization approach for the dynamic Capacitated Vehicle Routing Problem for Flash Flood Victim's Relief Operations
Florent Dubois, Patricia Stolf and Paul Renaud-Goud. *ROADEF 2022: 23ème congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*
2022 - Accepted

Capacitated Vehicle Routing Problem under Deadlines: An application to flooding crisis
Florent Dubois, Patricia Stolf and Paul Renaud-Goud. *IEEE Access*
2022 - Submitted

Dynamic Capacitated Vehicle Routing Problem for Flash Flood Victim's Relief Operations
Florent Dubois, Patricia Stolf and Paul Renaud-Goud. *International Conference on Information Systems for Crisis Response and Management (ISCRAM)*
2022 - Accepted

Bibliography

[1] H. Afsar, Christian Prins, and Andréa Santos. Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. International Transactions in Operational Research, 2013.

[2] H. Alaeddine, K. Serrhini, M. Maizia, and E Néron. A spatiotemporal optimization model for the evacuation of the population exposed to flood hazard. Natural Hazards and Earth System Science, 15(3):687 – 701, 2015.

[3] Maria Albareda-Sambola, Elena Fernández, and Gilbert Laporte. The dynamic multiperiod vehicle routing problem with probabilistic information. Computers & Operations Research, 48:31–39, 2014.

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

[5] Enrico Angelelli, Nicola Bianchessi, Renata Mansini, and Maria Gracia Speranza. Short term strategies for a dynamic multi-period routing problem. Transportation Research Part C: Emerging Technologies, 17(2):106 – 119, 2009.

[6] Apache. Apache activemq home page, 2004. URL `www.activemq.apache.org/`.

[7] Claudia Archetti, Elena Fernández, and Diana L. Huerta-Muñoz. The Flexible Periodic Vehicle Routing Problem. Computers & Operations Research, 85:58–70, 2017.

[8] Claudia Archetti, Elena Fernández, and Diana L. Huerta-Muñoz. A two-phase solution algorithm for the Flexible Periodic Vehicle Routing Problem. Computers & Operations Research, 99:27–37, 2018.

[9] Claudia Archetti, Dominique Feillet, Andrea Mor, and Maria Gracia Speranza. Dynamic traveling salesman problem with stochastic release dates. European Journal of Operational Research, 280(3):832 – 844, 2020.

[10] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. European Journal of Operational Research, 218(1): 1–6, 2012.

[11] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan, editors, Cloud Computing and Services Science, volume 367 of Communications in Computer and Information Science, pages 3–20. Springer International Publishing, 2013.

[12] Frédérique Baniel. Prise en compte d'objectifs de stabilité pour l'organisation de collectes de déchets. PhD thesis, 2009. Thèse de doctorat dirigée par Vidal, Thierry et Huguet, Marie-José Systèmes industriels Toulouse, INPT 2009.

[13] E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. Networks, 4(1):65–94, 1974.

[14] Frederick Benaben, Audrey Fertier, Aurelie Montarnal, Nicolas Salatge, Sébastien Truptil, Sebastien Rebiere Pouyade, and Matthieu Lauras. A Conceptual Framework and a Suite of Tools to Support Crisis Management. In HICSS, editor, HICSS 2017 - 50th Hawaii International Conference on System Sciences, pages p.237–246, Hawaii, United States, 2017. HICSS.

[15] Russell Bent and Pascal Van Hentenryck. Dynamic Vehicle Routing with Stochastic Requests. 2003.

[16] Russell W. Bent and Pascal Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. Operations Research, 52(6):977–987, 2004.

[17] Gerardo Berbeglia, Jean-François Cordeau, and Gilbert Laporte. Dynamic pickup and delivery problems. European Journal of Operational Research, 202(1):8–15, 2010.

[18] Djamel Berkoune, Jacques Renaud, Monia Rekik, and Angel Ruiz. Transportation in disaster response operations. Socio-Economic Planning Sciences, 46(1):23–32, 2012. Special Issue: Disaster Planning and Logistics: Part 1.

[19] Nicola Bianchessi and Giovanni Righini. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. Computers & Operations Research, 34(2):578–594, 2007.

[20] P. Bubeck, W.J.W. Botzen, H. Kreibich, and J.C.J.H. Aerts. Detailed insights into the influence of flood-coping appraisals on mitigation behaviour. Global Environmental Change, 23(5):1327–1338, 2013.

[21] Marina Campolo, Paolo Andreussi, and Alfredo Soldati. 1999.

[22] Elyn Charris, Christian Prins, and Andréa Santos. Solving the bi-objective Robust Vehicle Routing Problem with uncertain costs and demands. RAIRO - Operations Research, 50, 2016.

[23] Lichun Chen and Elise Miller-Hooks. Optimal team deployment in urban search and rescue. Transportation Research Part B: Methodological, 46 (8):984–999, 2012.

[24] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 1:89–101, 2003.

[25] George Dantzig and J. H. Ramser. The truck dispatching problem. Management Science, 6(1):80–91, 1959.

[26] Samuel Debionne, Isabelle Ruin, Saif Shabou, Céline Lutoff, and Jean Creutin. Assessment of commuters' daily exposure to flash flooding over the roads of the gard region, france. Journal of Hydrology, 541, 2016.

[27] Stéphanie Defossez, Tony Rey, Vinet Freddy, and Laurent Boissier. Flood risk management: cases studies in french mediterranean area. E3S Web of Conferences, 7:20019, 2016.

[28] Google Developers. Protocol buffers home page, 2001. URL `www.developers.google.com/protocol-buffers`.

[29] Edsger w. Dijkstra, Laurent Beauguitte, and Marion MAISONOBE. E.W. Dijkstra, 1959, A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1, p. 269271 Version bilingue et commentée. 2021.

[30] Johnny Douvinet. Flash flood hazard assessment in small agricultural basins coupling gis-data and cellular automata modelling. International Journal of Agricultural and Environmental Information Systems, 5:59–80, 2016.

[31] Moshe Dror and Pierre Trudeau. Split delivery routing. Naval Research Logistics (NRL), 37(3):383–402, 1990.

[32] Harald Dyckhoff. A typology of cutting and packing problems. European Journal of Operational Research, 44(2):145–159, 1990.

[33] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering, 57(4):1472–1483, 2009.

[34] Alan L. Erera, Juan C. Morales, and Martin Savelsbergh. The Vehicle Routing Problem with Stochastic Demand and Duration Constraints. Transportation Science, 44(4):474–492, 2010. Publisher: INFORMS.

[35] Aurélie Escudier, Pierre-Adrien Hans, Christophe Astier, and Jean-Luc Souldadié. From high waters forecasts to flooded areas forecasts. E3S Web of Conferences, 7:18008, 2016.

[36] Audrey Fertier, Anne-Marie Barthe-Delanoë, Aurelie Montarnal, Sébastien Truptil, and Frederick Benaben. Adoption of big data in crisis management toward a better support in decision-making. In 13th International Conference on Information Systems for Crisis Response and Management, page 7 p. ISCRAM, 2016.

[37] Marshall L. Fisher and Ramchandran Jaikumar. A generalized assignment heuristic for vehicle routing. Networks, 11(2):109–124, 1981.

[38] M.L. Fisher and R. Jaikumar. A Decomposition Algorithm for Large-scale Vehicle Routing. Department of Decision Sciences, Wharton School, University of Pennsylvania, 1978.

[39] Merrill M. Flood. The Traveling-Salesman Problem. Operations Research, 4(1):61–75, 1956. Publisher: INFORMS.

[40] Michael R Garey and David S Johnson. Computers and intractability, volume 29. W.H. Freeman New York, 2002.

[41] Michel Gendreau, Gilbert Laporte, and René Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transportation Science, 29(2):143–155, 1995.

[42] Michel Gendreau, François Guertin, Jean-Yves Potvin, and Éric Taillard. Parallel tabu search for real-time vehicle routing and dispatching. Transportation Science, 33(4):381–390, 1999.

[43] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. Parallel Computing, 27(12):1641–1653, 2001. Applications of parallel computing in transportation.

[44] Michel Gendreau, François Guertin, Jean-Yves Potvin, and René Séguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. Transportation Research Part C: Emerging Technologies, 14(3):157–174, 2006.

[45] Fred Glover. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 13(5):533–549, 1986.

[46] Bruce Golden and Arjang Assad. Vehicle routing with time window constraints. American Journal of Mathematical and Management Sciences, 6 (3-4):251–260, 1986.

[47] Bruce Golden, Arjang Assad, Larry Levy, and Filip Gheysens. The fleet size and mix vehicle routing problem. Computers & Operations Research, 11(1):49 – 66, 1984.

[48] R. L. Graham. Bounds for certain multiprocessing anomalies. The Bell System Technical Journal, 45(9):1563–1581, 1966.

[49] Léo Grange. Datacenter management for on-site intermittent and uncertain renewable energy sources. PhD thesis, 2019. Thèse de doctorat dirigée par Stolf, Patricia et Da Costa, Georges Sureté de logiciel et calcul de haute performance Toulouse 3 2019.

[50] Dani Hadad, Zeina Al Masry, Jean-Marc Nicod, Christophe Varnier, and Noureddine Zerhouni. A predictive decision approach for firefighters using variable selection technique. In 2019 Prognostics and System Health Management Conference (PHM-Paris), pages 258–263, 2019.

[51] Lars M. Hvattum, Arne Løkketangen, and Gilbert Laporte. Solving a Dynamic and Stochastic Vehicle Routing Problem with a Sample Scenario Hedging Heuristic. Transportation Science, 40(4):421–438, 2006.

[52] Lars Magnus Hvattum, Arne Løkketangen, and Gilbert Laporte. A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. Networks, 49(4):330–340, 2007.

[53] IBM. Cplex home page. URL `www.ibm.com/analytics/cplex-optimizer`.

[54] Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin. Exploiting Knowledge About Future Demands for Real-Time Vehicle Dispatching. Transportation Science, 40(2):211–225, 2006. Publisher: INFORMS.

[55] David S. Johnson. Near-optimal bin packing algorithms. 1973.

[56] Arun Jotshi, Qiang Gong, and Rajan Batta. Dispatching and routing of emergency vehicles in disaster mitigation using data fusion. Socio-Economic Planning Sciences, 43(1):1–24, 2009.

[57] Gwenaël Jouannic and Régis Thepot. How to "build back better" after a flood disaster? 2015.

[58] Heidi Kreibich, Philip Bubeck, Mathijs van Vliet, and Hans Moel. A review of damage-reducing measures to manage fluvial flood risks in a changing climate. 2015.

[59] Loïc Lannelongue, Jason Grealey, and Michael Inouye. Green algorithms: Quantifying the carbon footprint of computation. Advanced Science, 8 (12):2100707, 2021.

[60] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. European journal of operational research, 59(3): 345–358, 1992.

[61] Homero Larrain, Leandro C. Coelho, and Alejandro Cataldo. A Variable MIP Neighborhood Descent algorithm for managing inventory and distribution of cash in automated teller machines. Computers & Operations Research, 85:22 – 31, 2017.

[62] Homero Larrain, Leandro C. Coelho, Claudia Archetti, and M. Grazia Speranza. Exact solution methods for the multi-period vehicle routing problem with due dates. Computers & Operations Research, 110:148–158, 2019.

[63] Allan Larsen. The dynamic vehicle routing problem. PhD thesis, Technical University of Denmark, 2000.

[64] Allan Larsen, Oli B.G. Madsen, and Marius M. Solomon. Classification Of Dynamic Vehicle Routing Systems. In Vasileios Zeimpekis, Christos D. Tarantilis, George M. Giaglis, and Ioannis Minis, editors, Dynamic Fleet Management: Concepts, Systems, Algorithms & Case Studies, pages 19–40. Springer US, 2007.

[65] Jan Karel Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. Networks, 11(2):221–227, 1981.

[66] John D. C. Little, Katta G. Murty, Dura W. Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. Operations Research, 11(6):972–989, 1963.

[67] Chung Lang Liu and James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the ACM, 20(1):46–61, 1973.

[68] Gregor Lämmel and K Nagel. Multi agent based large-scale evacuation simulation. 2009.

[69] Snežana Mitrović-Minić, Ramesh Krishnamurti, and Gilbert Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. Transportation Research Part B: Methodological, 38(8):669–685, 2004.

[70] Babak Farhang Moghaddam, Rubén Ruiz, and Seyed Jafar Sadjadi. Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm. Computers & Industrial Engineering, 62(1):306–317, 2012.

[71] Gurobi Optimization. Gurobi home page. URL `www.gurobi.com/`.

[72] Siti Othman and Ghassan Beydoun. Metamodelling approach towards a disaster management decision support system. pages 614–621, 2010.

[73] Siti Othman and Ghassan Beydoun. Metamodel-based decision support system for disaster management. volume 2, pages 403–412, 2010.

[74] Puca Huachi Vaz Penna, Andréa Cynthia Santos, and Christian Prins. Vehicle routing problems for last mile distribution after major disaster. Journal of the Operational Research Society, 69(8):1254–1268, 2017.

[75] Jean-Marc Pierson, Gwilherm Baudic, Stéphane Caux, Berk Celik, Georges Da Costa, Léo Grange, Marwa Haddad, Jérôme Lecuivre, Jean-Marc Nicod, Laurent Philippe, Veronika Rehn-Sonigo, Robin Roche, Gustavo Rostirolla, Amal Sayah, Patricia Stolf, Minh-Thuyen Thi, and Christophe Varnier. Datazero: Datacenter with zero emission and robust management using renewable energy. IEEE Access, 7:103209–103230, 2019.

[76] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L. Medaglia. A review of dynamic vehicle routing problems. European Journal of Operational Research, 225(1):1–11, 2013.

[77] Olivier Piller, Fereshte Sedehizade, Thomas Bernard, Mathias Braun, Nicolas Cheifetz, Jochen Deuerlein, Andreas Korth, Emmanuel Lapébie, Iris Trick, Jean-Marc Weber, and Caty Werey. Resiwater: A franco-german project for augmented resilience of water distribution systems following severe abnormal events. 2016.

[78] Jean-Yves Potvin, Ying Xu, and Ilham Benyahia. Vehicle routing and scheduling with dynamic travel times. Computers & Operations Research, 33(4):1129–1137, 2006.

[79] Jennifer K. Poussin, W.J. Wouter Botzen, and Jeroen C.J.H. Aerts. Factors of influence on flood damage mitigation behaviour by households. Environmental Science & Policy, 40:69–77, 2014.

[80] Sigrid Johansen Rennemo, Kristina Fougner Rø, Lars Magnus Hvattum, and Gregorio Tirado. A three-stage stochastic facility routing model for disaster response planning. Transportation Research Part E: Logistics and Transportation Review, 62:116–135, 2014.

[81] Ulrike Ritzinger, Jakob Puchinger, and Richard F. Hartl. A survey on dynamic and stochastic vehicle routing problems. International Journal of Production Research, 54(1), 2016.

[82] Inmaculada Rodríguez-Martín, Juan-José Salazar-González, and Hande Yaman. The periodic vehicle routing problem with driver consistency. European Journal of Operational Research, 273(2):575–584, 2019.

[83] Samuel Rufat, Eric Tate, Christopher G. Burton, and Abu Sayeed Maroof. Social vulnerability to floods: Review of case studies and implications for measurement. International Journal of Disaster Risk Reduction, 14:470–486, 2015.

[84] R. Russell and W. Igo. An assignment routing problem. Networks, 9(1): 1–17, 1979.

[85] Celso Sakuraba, Andréa Santos, Christian Prins, Lucie Bouillot, Arnaud Durand, and Bernard Allenbach. Road network emergency accessibility planning after a major earthquake. EURO Journal on Computational Optimization, 4, 2016.

[86] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. Transportation Science, 29(1):17–29, 1995.

[87] Guy Schumann, Renaud Hostache, Christian Puech, Lucien Hoffmann, Patrick Matgen, Florian Pappenberger, and Laurent Pfister. High-resolution 3-d flood information from radar imagery for flood hazard management. IEEE Transactions on Geoscience and Remote Sensing, 45(6): 1715–1725, 2007.

[88] A.K Schwab, D Sandler, and D.J Brower. Hazard Mitigation and Preparedness: An Introductory Text for Emergency Management and Planning Professionals. CRC Press, 2016.

[89] Nicola Secomandi and François Margot. Reoptimization approaches for the vehicle-routing problem with stochastic demands. Operations Research, 57(1):214–230, 2009.

[90] Leonard Shabman and Kurt Stephenson. Searching for the correct benefit estimate: Empirical evidence for an alternative perspective. Land Economics, 72(4):433–449, 1996.

[91] Jiuh-Biing Sheu. An emergency logistics distribution approach for quick response to urgent relief demand in disasters. Transportation Research Part E: Logistics and Transportation Review, 43(6):687–709, 2007. Challenges of Emergency Logistics Management.

[92] Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research, 35(2):254–265, 1987.

[93] Gheorghe Stancalie, Vasile Craciunescu, and Anisoara Irimescu. Development of a downstream emergency response service for flood and related risks in romania based on satellite data. E3S Web of Conferences, 7:17007, 2016.

[94] Patricia Stolf, Julian Berthet, Christelle Bosc, Pauline Bremond, Arnaud Ceyte, Victor Champonnois, Anne-Laure Collard, Florent Dubois, Georges Da Costa, Katrin Erdlenbruch, Didier Felts, Rémy Gasset, Frédéric Grelot, Christophe Héral, Benjamin Piccinini, Jean-Marc Pierson, and Amal Sayah. Towards Flash Floods Management Using Temporal Loops. Research Report IRIT/RR–2021–06–FR, IRIT - Institut de Recherche en Informatique de Toulouse, 2021.

[95] Ilgaz Sungur, Fernando Ordóñez, and Maged Dessouky. A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. IIE Transactions, 40(5):509–523, 2008.

[96] Yves Tramblay and Samuel Somot. Future evolution of extreme precipitation in the mediterranean. Climatic Change, 151(1):289–302, 2018.

[97] Pascal Van Hentenryck, Russell Bent, and Eli Upfal. Online stochastic optimization under time constraints. Annals of Operations Research, 177 (1):151–183, 2010.

[98] Rinde R. S. VanLon, Eliseo Ferrante, Ali E. Turgut, Tom Wenseleers, Greet Vanden Berghe, and Tom Holvoet. Measures of dynamism and urgency in logistics. European Journal of Operational Research, 253(3): 614–624, 2016.

[99] Freddy Vinet, D. Lumbroso, Stéphanie 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: Journal of the International Society for the Prevention and Mitigation of Natural Hazards, 61(3):1179–1201, 2012.

[100] Duc Minh Vu, Mike Hewitt, Natashia Boland, and Martin Savelsbergh. Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. Transportation Science, 54(3):703–720, 2020.

[101] Min Wen, Jean-François Cordeau, Gilbert Laporte, and Jesper Larsen. The dynamic multi-period vehicle routing problem. Computers & Operations Research, 37(9):1615–1623, 2010.

[102] Sascha Wohlgemuth, Richard Oloruntoba, and Uwe Clausen. Dynamic vehicle routing with anticipation in disaster relief. Socio-Economic Planning Sciences, 46(4):261–271, 2012. Special Issue: Disaster Planning and Logistics: Part 2.

[103] Jian Yang, Patrick Jaillet, and Hani Mahmassani. Real-Time Multivehicle Truckload Pickup and Delivery Problems. Transportation Science, 38(2): 135–148, 2004.

[104] Wen-Huei Yang, Kamlesh Mathur, and Ronald H. Ballou. Stochastic vehicle routing problem with restocking. Transportation Science, 34(1): 99–112, 2000.

[105] Lin Zhu, Louis-Martin Rousseau, Walter Rei, and Bo Li. Paired cooperative reoptimization strategy for the vehicle routing problem with stochastic demands. Computers & Operations Research, 50:1–13, 2014.