Dynamic rerouting in multimodal transportation networks

Omar Dib

SPIM

Thèse de Doctorat

école doctorale sciences pour l'ingénieur et microtechniques
UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

n°ordre 015

REROUTAGE DYNAMIQUE DES PASSAGERS DANS LES RÉSEAUX DE TRANSPORT MULTIMODAUX

Omar DIB

Soutenue le 06/11/2017 à 14:00 à Belfort

Jury

Rapporteur	Nadia BRAUNER, Professeur, Université de Grenoble Alpes
Rapporteur	Xavier GANDIBLEUX, Professeur, Université de Nantes
Examinateur	Lionel SCREMIN, Chef de projet, Alstom Transport
Examinateur	Jakob PUCHINGER, Professeur, LGI – CentraleSupélec
Directeurs de thèse	Alexandre CAMINADA, Professeur, Polytech Nice Sophia
	Marie-Ange MANIER, Maître de Conférences HDR, UTBM
	Laurent MOALIC, Maître de Conférences, UHA

OPERA
OPtimisation Et RéseAux

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ DE BOURGOGNE FRANCHE COMTÉ

Spécialité

Informatique

Ecole doctorale Sciences Pour l'Ingénieur et Microtechniques (SPIM)

Présentée par

Omar DIB

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ de BOURGOGNE FRANCHE COMTÉ (UBFC)

Sujet de la thèse :

Re-routage dynamique des passagers dans les réseaux de transport multimodaux

Soutenue le 06-11-2017 à 14 :00 à Belfort

devant le jury composé de :

Mme. Nadia BRAUNER	Rapporteur	Université Grenoble Alpes
M. Xavier GANDIBLEUX	Rapporteur	Université de Nantes
M. Jakob PUCHINGER	Examinateur	CentraleSupélec
M. Lionel SCREMIN	Examinateur	Alstom Transport
Mme. Marie-Ange MANIER	Co-directrice	UTBM
M. Laurent MOALIC	Co-encadrant	UHA
M. Alexandre CAMINADA	Directeur de thèse	Polytech Nice Sophia

DOCTOR OF SCIENCE THESIS
BOURGOGNE FRANCHE-COMTÉ UNIVERSITY

Specialization

Computer science

Doctoral School Engineering Sciences and Microtechnologies

Presented by

Omar DIB

For the degree of

DOCTOR OF SCIENCE OF BOURGOGNE FRANCHE-COMTÉ UNIVERSITY

PhD thesis subject :

Dynamic rerouting of passengers in multimodal transportation networks

Defended on 06-11-2017 at 14 :00

Committee in charge :

Mme. Nadia BRAUNER	Rapporteur	Université Grenoble Alpes
M. Xavier GANDIBLEUX	Rapporteur	Université de Nantes
M. Jakob PUCHINGER	Examinateur	CentraleSupélec
M. Lionel SCREMIN	Examinateur	Alstom Transport
Mme. Marie-Ange MANIER	Co-encadrante	UTBM
M. Laurent MOALIC	Co-encadrant	UHA
M. Alexandre CAMINADA	Directeur de thèse	Polytech Nice Sophia

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor, Professor Alexandre CAMINADA for his continuous support, guidance, and encouragement throughout this thesis. This work would not be successful without his valuable advice, pertinent ideas and immense knowledge.

Special thanks are given to my associate advisors, the assistant professors Marie-Ange MANIER and Laurent MOALIC, as well as the MIC Project Leader Lionel SCREMIN, for their guidance and assistance in the completion of this thesis. I could not have imagined having better advisors and mentors for my Ph.D study.

I would also like to thank my thesis committee members for all their guidance through this process. Their insightful comments, ideas, questions and feedback helped improve the quality of this manuscript and incented me to widen my research from various perspectives.

My sincere thanks also go to all ALSTOM members in the MIC project, in particular the Innovation Program Manager Pascal POISSON, and the Multimodal Supervision Expert Manal ABID. Their long term vision, practical advice, and continuous assessment have helped a lot to finish this thesis.

I thank all my colleagues at IRT SystemX for the stimulating discussions, for the long time we were working together before deadlines, and for all the fun we have had in the last three years.

Last but not the least, I would like to express my gratitude to my parents, brothers and sisters for their unconditional support and encouragement.

This research work has been carried out in the framework of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program "Investissements d'Avenir".

Omar DIB

Belfort, 30 July 2017

Abstract

The human mobility is nowadays organized in a multimodal context with more and more complex transport networks. The number of passengers is increasing and new transport modes enter the system day after day simultaneously with new mobility behaviors. As a result, users usually find themselves more confused when choosing between several possibilities to go from one place to their destination. For the sake of helping them to efficiently navigate through this intricate transportation scheme, an efficient Travelers Information System (TIS) has to be built. Thanks to such a system, the transport operators seek not only to provide passengers with optimal itineraries, but also with efficient and reliable solutions in case of disruptions.

In fact, commuters do not only seek short time travels, but they usually consider several other criteria such as comfort and effort. An efficient routing system should therefore take into account the various needs and preferences of each passenger. Besides, transport modes are often prone to delays. Thus, handling uncertainty is also a very important aspect of practical journey planning systems. Moreover, the proposed multimodal routes should not only be feasible in a static case, but also robust against the dynamic and stochastic variations of the transport system. Furthermore, crucial constraints should be taken into account such as the capacity limitation of vehicles and the time complexity of the developed routing algorithms.

The main objective of this thesis is to propose a formulation that adequately allows representing a multimodal network. Based on our formulation, we elaborate several efficient routing algorithms. In particular, we focus on solving the Earliest Arrival Problem in a single/multiple criteria context, both in dynamic and stochastic environments. To deal with the real time complexity issue, metaheuristics such as Genetic Algorithms (GA), Variable Neighborhood Search (VNS) and Memetic Algorithms (MA) have been used.

The computational performance of this work has been assessed by developing a real world route planning system, and solving real life itinerary planning problems defined on the transport network of the French Region Île-de-France that includes the city of Paris and its suburbs. The emerging computational results indicate that the numerous basic and complex instances were solved within a reasonable amount of time and the integration of the proposed routing framework as a module of an operational TIS is relevant.

Keywords— Multimodal route planning, modelling, solving, genetic algorithms, variable neighbourhood search, memetic algorithms, real world application.

Résumé

La mobilité humaine s'organise de nos jours dans un contexte multimodal avec des systèmes de transport particulièrement complexes. Le nombre d'usagers va croissant et de nouveaux modes de transport émergent jour après jour en même temps que de nouveaux comportements de mobilité. En conséquence, les utilisateurs se trouvent généralement confrontés à la nécessité de choisir entre plusieurs possibilités pour se rendre d'un point d'origine à leur lieu de destination. Dans le but de les aider à naviguer facilement à travers ces réseaux complexes, un système efficace d'information voyageurs doit être construit. Grâce à ce système, les opérateurs de transport cherchent non seulement à fournir des itinéraires optimaux, mais aussi des solutions efficaces et fiables en cas de perturbations.

En réalité, les passagers ne cherchent pas seulement à minimiser le temps de trajet. Ils ont aussi tendance à considérer d'autres critères tels que le confort et l'effort qu'il s'agit alors d'optimiser. Un système de routage efficace devrait donc tenir compte des préférences de chaque passager. En outre, les modes de transport sont souvent enclins à des retards. La prise en considération de l'incertitude est donc un aspect très important pour un calculateur d'itinéraires. Les itinéraires multimodaux proposés doivent non seulement être réalisables dans un cas statique, mais également suffisamment robustes face à des aléas. En outre, tout en intégrant des contraintes essentielles telles que la capacité limitée des véhicules, une attention particulière devrait aussi être accordée à la complexité temporelle des algorithmes de routage développés.

L'objectif principal de cette thèse est d'élaborer une formulation qui permet de représenter adéquatement un réseau de transport multimodal. En nous basant sur cette formulation, nous proposons plusieurs algorithmes de routage. Nous nous concentrons en particulier sur la résolution de problèmes de plus courts chemins dans un contexte mono/multicritère dans un réseau de transport dynamique et stochastique. Des métaheuristiques telles que les Algorithmes Génétiques (GA), la méthode de Recherche à Voisinage Variable (VNS) et les Algorithmes Mémétiques (MA) ont été utilisés pour fournir des résultats en temps réel.

La performance de calcul a été évaluée en résolvant des requêtes de routage à l'échelle du réseau de transport de la Région française Île-de-France. Les résultats indiquent que les nombreuses instances traitées ont été résolues dans un laps de temps raisonnable et que nos algorithmes de routage sont suffisamment performants pour être intégrés dans un calculateur d'itinéraire multimodal opérationnel.

Mots-clés— Routage multimodal, modélisation, résolution, algorithmes génétiques, recherche à voisinage variable, algorithmes mémétiques, application pratique.

Table of contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

In this chapter, we first present the general context of this thesis. An emphasis is given to the Technological Research Institute IRT SystemX where this thesis took place. We focus also on the MIC (Modelling, Interoperability, Communication) project, in which this research was conducted. We then define the multimodal transportation system studied with its various components and utilities. Besides, we devote section 1.2 to highlight the various motivations behind this thesis. In section 1.3, we present our main contributions of this thesis. We finally give in section 1.4 a general overview of all chapters included in this manuscript.

Sommaire

Related publications:

1. My thesis in 3 minutes Watch Video
2. General poster about the thesis Download
3. A brief report about the thesis Download
4. Thesis midterm assessment Download

1.1 Context

A IRT SystemX

The Research Institute of Technology SystemX, dedicated to the future system digital engineering, is a powerful innovation engine for addressing the scientific and technological challenges of transport and mobility, communications, digital safety and energy markets. The involved academic and industrial research teams are colocated on the "Saclay Cluster" and share the same ambition: strengthening the “ Training-Research Industry ” momentum to generate major technology transfers and accelerate competitiveness, attractiveness and sustainability across companies and French business as a whole.

This newly Research & Technology Institute has been granted with 336 M€ as part of the French national investments for the future and has been certified by the SYSTEMATIC-PARIS-REGION competitiveness cluster. SystemX is also supported by local and regional authorities. The funding members are: Alstom, Renault, Bull, Kalray, Sherpa, OVH Global Solutions, Systematic Paris-Region, Inria, Institut Mines-Telecom and Campus Paris Saclay.

The IRT SystemX can so be seen as a federation of industrial and academic researchers located on the campus Paris Saclay. Current industrial partners of the IRT involved in the present multimodal project include ALSTOM, RENAULT, SNCF, INRIA, CEA and some technological small and medium-sized enterprises.

Key figures: 30 Research & Development projects, 70 partners, 1 training program dedicated to engineering systems, 300 researchers by 2020.

B MIC Project

MIC (Modeling Interoperability Communication) project is looking forward to enabling the sizing, the positioning, the optimization and the supervision of multimodal transport. As a first step forward, the project members provide herein a collective review of the state of the knowledge and capability in the relevant scientific and technological fields concerned by multimodal mobility. Once the current state of the art is understood, the project can identify the gaps in knowledge where specific research shall be conducted.

Multimodal transport evolves as a system of systems with numerous collective modes, growing number of individual vehicles, new technologies, and a thirst for individual mobility conflicting undermined limits in terms of capacity, performance and accessibility. Tackling such complexity requires proof of concepts relying on modelling and simulation. Hence, a genuine challenge lies in the appropriate mod-

elling. Specifically, several dynamic models have given efficient assessment of car traffic at a macroscopic level of a road. Yet, an extended version of such model to include other modes needs inquiring.

From a supervision point of view, road monitoring has set the first historical milestones in the transport area. It has been followed closely by railways supervision. Both of them evolved independently and built traditional standards for monomodal supervision. More recently, the combination of transport modes fostered the emergence of a multimodal traffic management. New technologies, new scientific research and increasing public and private project initiatives provide all together a perspective of the likely future requirements on multimodal supervision and optimization methods. In most cases, multimodal supervision cannot arise from ground zero but rather is bound to emerge from a cooperative approach across existing traffic management systems, hence a strong incentive for interoperability.

Finally, multimodality increases the number of stakeholders. Exploring the new legal and economical interactions is a field of research in itself. As such, the feasibility of multimodal transportation is tied up with the understanding of business models encompassing techno-legal concerns jointly with purely technical parameters. By the same token, electrical carsharing is an interesting case study. Station positioning, fleet sizing and vehicles balancing are technical objects closely coupled with social benefits (air pollution and noise reduction, land use for public interests rather than individual parking), economic incentives (in western countries, cars are nowadays rather seen as a service than a property) and contractual concerns (purchasing, renting, litigation and insurance).

Problems considered within this project, relevant to the proposed PhD, include:

1. The design of new transportation networks, to optimize criteria taking into account the service provided to the commuters and also the energy consumption (the goal being to advise decision makers in organizations like STIF2).

2. The microscopic and macroscopic modelling of road traffic (based on the expertise of IFSTTAR).

3. The rerouting of customers, exploiting recent information technologies, which provide a better monitoring of network congestion through cellular phones, and the possibility to provide in real time rerouting recommendations to the customer (*e.g.* in the occurrence of incident). Closely related issues arise in the development of computer aided decision tools for train line regulators (provided as component of information systems associated with Alstom solutions in particular) with a strong attention to scalability issues.

C Multimodal Transportation Networks

In this section, we introduce several definitions of multimodal transport networks. We also highlight the importance of having such systems in real world for both transport operators and users.We finally present the various components of a multimodal system, and we describe how interactions between them happen.

C.1 Definition

With the dawn of history, the first and single transportation mode was "walking". Hence, the transportation system only consisted of paths. However, with people discovering the possibility to use animals as a mean to travel or transfer goods from one place to another, a new mode of transport has been added to the whole system. Later on, different types of vehicles were added simultaneously. Thus, we started talking about a "Multimodal Transportation System" in which nodes and terminals have been added to the components of the initial transportation system.

The network always comprises a set of routes, where a route represents a single path between two points. Nodes and terminals are the contact or exchange points where it is possible for people to change from one mode to another.

As a complex system, the transport network, also called combined transport system, does not have a clear and unique definition. Indeed, several definitions have been identified, each depending on a point of view: whether it was the mode of transport, the infrastructures, the operators, or even the users [**?**].

Some authors define it as a collection of subsystems where each sub-system represents one mode of transport and has its own standards and rules. Others described it in terms of its components; [**?**] define it as "the assemblage of components associated with a specific means of transport".

In this thesis, a Multimodal Transport Network refers to the combination of all traveller modes and kinds of transportation systems operated through various systems. In another term, it is a set of choices of modes of transport that travellers can use simultaneously according to their needs in order to reach their destinations.

C.2 Importance

The "Multimodal Transportation System" originally gained its importance from the fact that the combination of several modes of transport into one large system would maximize users' profits.

Since each transport mode has both its pros and cons, the system tends to combine the advantages of several modes in an attempt to overcome the negatives of each mode. For instance, a bike's capacity in terms of travelled distance remains limited because of the limited speed and the associated physical effort. However, it is cheap, environmentally friendly, and can take its rider even through narrow roads. On the other hand, public transport such as a bus or a metro can be used for long-distance journeys, but at the same time, its direction is scheduled and its stations might be somehow distant.

Furthermore, multimodal transport can save passengers' time. For example, to go from one place to another one, a passenger can take a bus but it will take one hour. However, by combining several modes, the passenger can take one bus to the subway station, then a metro to reach his destination; it will finally take 30 minutes.

C.3 Components

After defining the multimodal transport system and discussing its importance, the components of the system are basic points to discuss also. Actually, the main components of the multimodal transportation system can be distinguished as: the modes of transport and their infrastructures from one side, and the movements of the travellers on the other side. The intersections of these components bring the concepts of transfer and transit [**?**].

1. **Modes of transport:** Thanks to the technological revolution, the transportation system has become more and more developed. Several modes of transport may be used nowadays, and they are either private or public.
   - Individual modes: They are mostly dedicated to private usage and they also offer continuous service at any time. They are also subdivided into motorized and non-motorized modes:
     - Motorized: automobile, motorcycle, taxi...
     - Non-motorized: bicycle, walking...
   - Collective modes: They are the opposite of individual modes. Actually, they are shared transport services which are available for the general public use. They usually offer discrete service according to pre-defined timetables. Collective transport modes include buses, trolleybuses, trams and trains, rapid transit (metro/subways/undergrounds etc.)
2. **Infrastructure:** As we discussed before, some authors described the transport system in terms of its infrastructures. They argue that infrastructures in a transportation system may take several forms: road, rail, water or air. They

added also that a single infrastructure can sometime support different modes. For instance, a road can serve public buses and private cars simultaneously.

3. **Transfer points:** They are the connection points between two or several modes of transport. In another term, they are the places where people have to change from one mode to another in order to reach their destination. They usually play an essential role in a multimodal transportation system.

It can be said now that the multimodal transportation system is an advanced version of the general transportation system. It can also be said that the system's facilities and components elaborate its importance in actual lives. However, we cannot ignore the fact that some services such as route planning should be added to the whole system to make it work more efficiently.

C.4 Current Route Planning Systems

Nowadays, several solutions exist in order to solve conventional route planning problems. These solutions come in the form of free or paid software and applications [**?**]. Mostly, their aim is to help in finding the shortest path between the passenger's origin and his destination (*e.g.* Google Maps, Bing Maps).

Generally, we can distinguish between individual route planners that deal with continuous modes of transport and collective route planners that consider discrete modes.

Individual route planners usually take into account individual transport modes which are qualified as continuous such as cars, bikes and walking. Globally, they are based on GPS technology in order to find a path between an origin and a destination (*e.g.* Google Maps, MyWays).

Such systems usually consider that passengers will only use one single continuous mode during their journeys. Consequently, they may not be able to find a path between any origin and destination places and in some case they may not provide the optimal path. Usually by using more than one mode of transport, we may reach more places in short time.

Disregarding collective transportation modes from any route planning system will require additional effort to simultaneously use several routing applications. Added to the extra effort, combining results of several routing applications may not always lead to obtain optimal solutions. Some of those applications do offer more than one transportation mode, but again they still work on each mode separately.

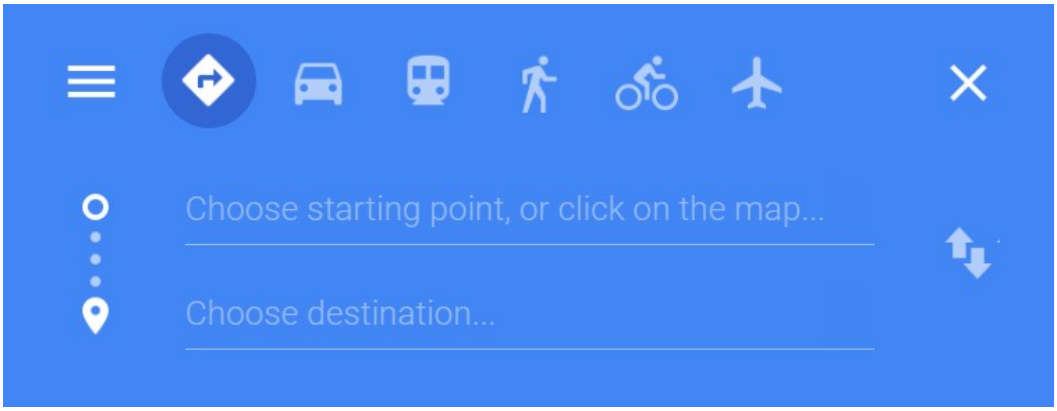


Figure 1.1 – Google Maps route planner

Figure 1.2 – GARMIN car navigator

In contrast, collective transit route planners, are dedicated specifically to collective modes of transportation (*e.g.* bus, subway...) that work according to predefined timetables. They calculate the paths between two points regarding the accessibility of the various means of transport [**?**]. Moreover, they are capable of mixing more than one mode in order to reach a destination (*e.g.* RATP in Paris). Regardless the efficiency of such route planners, they are still not able to cover the whole aspect of multimodal system. Indeed, they have the lack of considering some other continuous modes like taxi, cycle or car sharing.

Figure 1.3 – Multimodal route planner

For that, a single route planning system that integrates both individual and collective modes is believed to be capable of providing more and more accessible routes, as well as providing the optimal route using different modes of transport.

Besides, most current route planners claim considering the preferences of passengers. That is usually done by adding additional constraints to routing algorithms such as fixing an upper limit bound for transfer time or number of transfers. Also, preferences may be handled by adding some constraints to the transportation modes used along users' journeys. It is worth to mention that such consideration of users' preferences does not constitute a bottleneck for routing algorithms. However, it decreases the algorithm's search space and thus accelerates their running time.

The difficulty in handling users' preferences becomes critical whenever such preferences come in the form of multiple criteria to be optimised [**?**]. This usually refers to solving multiple criteria optimization problem. In the literature, solving such problems is very hard and requires a huge amount of running time especially when more than two criteria are considered.

To overcome such limitations, current route planners often use some simplifications such as considering criteria in a lexicographical order or transforming the problem to a trivial monocriteria optimization problem. The main drawback of such methods is that they may ignore very important compromise solutions. As a result, users may not always obtain efficient solutions.

Furthermore, current routing applications do not include predictions into their representation schemes or routing algorithms [**?**]. As a result, the proposed itineraries may not always reflect the reality. Thus, less robust solutions are provided.

Finally, most current route planners do not allow rerouting of passengers in case of disruptions. This is due to the incapacity of handling real time traffic information or the limitation of routing algorithms to integrate the vehicle capacity and other constraints inside their search process.

1.2 Motivations

In modern societies, transportation networks become more and more complex with the development of infrastructure construction for the purpose of facilitating mobility. However, high-complexity, high-density, multilayer and multimodal transportation may not automatically bring convenience to users. They may rather confuse user's planning while going from one place to another one.

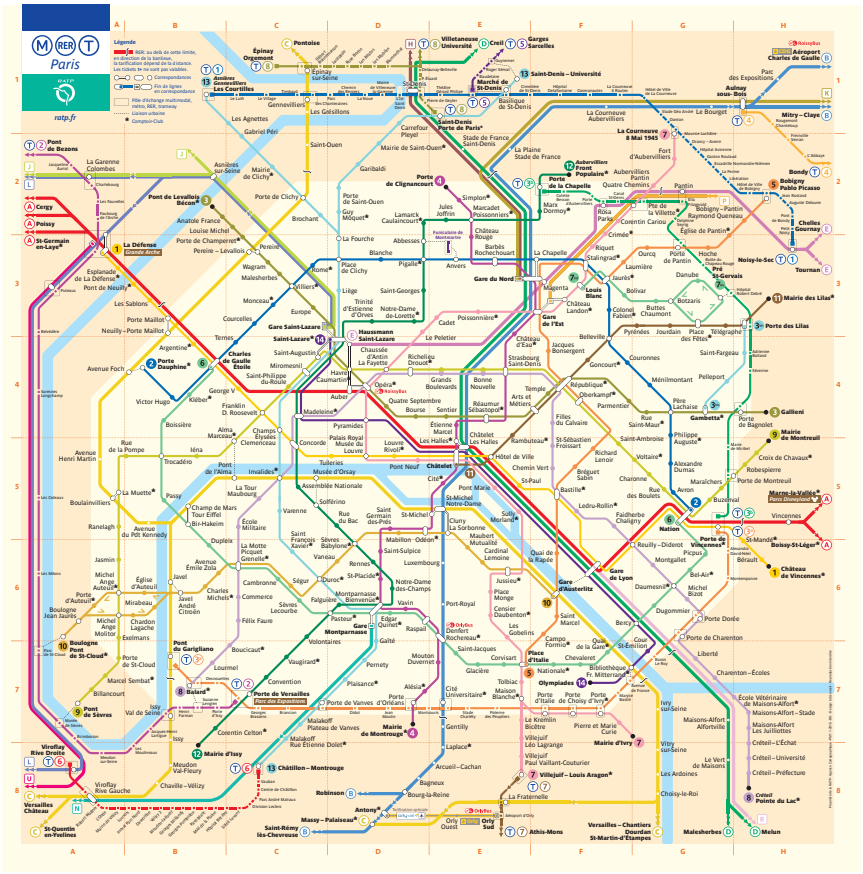


Figure 1.4 – Paris subway map

In Figure 1.4 we show a map with a brunch of rendered transportation networks. The density plotted lines with different styles and attached labels re interwined with each other. There is so much information contained in the highly developed transportation networks that many people may find it difficult to figure the best routes out. Consequently, for the sake of helping people efficiently to find the best paths through the transportation infrastructure, route planning is gaining more and more importance.

In recent years, personal navigation service is becoming more widespread. It benefits from the rapid development of supporting technologies including high-accuracy positioning, high speed data communication, powerful mobile computation, openness and standardization of navigation data, efficient data processing and analysis. Although a personal navigation service or application contains multiple navigation related functions, and has a complicated underlying workflow, it basically answers no more than three questions: where am I? where the destination is? and how to get there? Route planning, as an important function of the service, is responsible for a part of the third question based on the results of the first two. Such shot of service is available in some commercial product such as in GPS systems, vehicle navigation devices or even smart phones, which have been populated with navigation functions.

In real life, people may find difficulties when they try to plan routes in a complex transportation network across multiple different transport means with the above mentioned products or systems. For instance, a car navigation does not work in collective transit system. And a collective transit route planner cannot tell a user who has a car or bicycle available on departure and how to find feasible station with such individual traffic tools. As a result, there is a real need to develop a seamless application by which the various components and subsystems included in a transportation system are considered. Individual transport systems such as car, bike and pedestrian have to be integrated with collective transport modes in the goal of providing a real multimodal service.

Besides, most current routing applications disregard some crucial properties of the transport system. This latter is not a static network; it, however, changes with time. The state of the system in the morning is not the same as its state at noon or at night. Such a dynamic is also originated from the fact that some services are only available at specific time along the day. Ignoring the dynamic aspect in a navigation system may lead to imprecise or even incorrect itineraries. Consequently, handling the time dependency in multimodal networks is one point that motivated us to accomplish this research.

Analysing current routing systems has also guided this work towards studying stochastic route planning issues. Indeed, the stochasticity aspect of transport networks is almost disregarded in all modern routing applications. This property is essential since it directly affects the quality of services provided to users. When proposing a route, it is not only important to compute its travel time in an ideal case, but it is also crucial to provide information about its variations in case of disruptions. Therefore, computing routes in stochastic environments, as well as providing passengers with more robust itineraries, is another element that motivates this work.

As we discussed before, people are nowadays more susceptible to use several

modes of transport in their travels. However, the usage of those modes will undoubtedly be altered as it might be affected by a set of circumstances embodied in the form of disturbances and perturbations. Generally, in transportation system, there exist operators who control parts of the network (*e.g.* SNCF, RATP, Albatrans in Paris). There are also organizations that monitor the quality of services provided by operators to passengers (*e.g.* STIFF in Paris). Mostly, operators inform passengers about the traffic, the accessibility of the various modes of transport and the services times. Therefore, commuters depend on the timetables and schedules set by the operators in order to plan their travels accurately. However, in real situations, maintaining an ideal service is somehow impossible since many factors can easily affect the system's anticipated run. Thus, there is always a need to have alternative solutions that deal with the abnormal case.

Moreover, we have noticed that most routing applications claim offering services with respect to passengers' preferences. However, that is usually done by performing a single criterion optimization or by solving multiple criteria problems using trivial methods. As a result, users are not always provided with all optimal solutions while accomplishing their travels in the Pareto context [**?**].

In fact, in large cities like Paris where the transport system comprises millions of passengers, and thousands of vehicles, building such a magical system with zero-failure probability is an impossible mission. Therefore, providing customized and effective alternatives to the passengers is now a big challenge for city agencies. To overcome that, we propose in this work a general framework to solve multiple criteria routing issues in a static, dynamic or even stochastic multimodal networks.

Moreover, while rerouting passengers, a special attention should be given to the capacity of vehicles. A railway with 1,000 passengers cannot be rerouted to a small bus with a capacity of 50. Integrating capacities in the modelling approach, as well as in routing algorithms, have to be done in order to provide high quality services. By analysing the current state of routing systems, it can be noticed that the capacity aspect is not dealt with. Therefore and in the goal of intelligently manage disruptions, we aim at proposing efficient supervision strategies.

Finally, our analysis of the state of the art of traditional shortest path algorithms has shown that some of them cannot deal with multiple criteria problems. Others are not well suited for dynamic and stochastic environment. The limitations may be due to the high computational time that prevents them from being used in real world contexts, or the flexibility to deal with additional problem constraints. Therefore, in this thesis, we propose several contributions whereby basic and advanced routing issues are efficiently solved.

1.3 Main Contributions

The main contributions of this PhD are presented in this section.

A Modelling Approach

We first propose a modelling approach for representing a multimodal transportation network. Both individual transport modes that offer continuous service such as pedestrian, car and bike and collective transport modes that work according to predefined timetable such as bus, tram and metro are taken into consideration. A subdirected graph is introduced for every transport modes and the pedestrian network, as well as, transfer edges are used to link all networks together. A multimodal graph is then obtained whereby computing multimodal routes becomes feasible. A special attention is given to the modelling of platforms inside stations in a collective transport mode. The presence of several access points to one station is also considered while modelling stations. To account for the dynamic aspect of the transport system, time dependent edges are used. The weight of each edge depends on the time at which the edge is crossed. Also, the weight of any edge may be static, dynamic or stochastic. The proposed approach also allows dynamically considering the vehicle's capacities of the various transport modes. This is very useful while rerouting passengers in case of disruptions. Finally, other issues are taken into consideration such as the presence of escalators or other services dedicated to persons with reduced mobility.

B Route Planning with Exact Algorithms

After the modelling approach, we propose an exact routing algorithm for computing the set of Pareto optimal solutions to go from one place to another. Both individual and collective transportation modes are taken into consideration. Several criteria are optimized such as the travel time, the number of transfers, the walking time and the cost of a journey. Besides, we propose several accelerating techniques so that the search space of the proposed algorithm is reduced.

C Route Planning with Metaheuristics

We show that the computational complexity of the exact approach constitutes a bottleneck for the algorithm to be used in real world route planning system. Therefore, we propose solving routing issues with metaheuristics. High quality solutions are obtained within a reasonable amount of computational time. The proposed approach is a combination between Genetic Algorithm that belongs to the population

based metaheuristics and Variable Neighbourhood Search that belongs to the single solution metaheuristics.

D Stochastic Route Planning

We also study in this paper the stochastic route planning issue. We show that the quality of the proposed routes may become very poor due to small variations in the transport system. Therefore, we propose an approach so that the weight of an edge is a stochastic variable that depends on time. By doing so, we provide users with the distribution of the arrival time of the provided routes. Besides, we study the multicriteria route planning in a stochastic environment. A new algorithm is proposed where the best, average and worse case of the arrival time and the standard deviation are considered as criteria to be optimized. The results obtained show that robust solutions can be obtained by integrating those criteria in the routing algorithm.

E Web-Based Routing Application

To assess the performance of the proposed modelling approach, as well as routing algorithms, a web-based routing application has been developed. Data used belongs to the French Region Ile-de-France that includes the city of Paris and its suburbs. We present in this thesis the various data structures used during implementation and the various client-server components of the routing application.

1.4 Overview

This thesis is organized as follows: In Chapter 2, we present a brief state of the art on various routing issues and several well known shortest path algorithms. We also present some metaheuristics and their applications in the transportation field. In Chapter 3, we introduce the modelling approach of multimodal transportation networks, as well as a general mathematical formulation of various routing issues. In Chapter 4, we present several routing algorithms that have been developed for solving mono/multicriteria route planning problems. Chapter 5 is devoted to explain how different metaheuristics are adapted to solve basic and advanced routing problems. In Chapter 6, we consider that the route planning problem is a stochastic multimodal networks. We show how the stochasticity is taken into consideration and how stochastic routing algorithms are designed. Each part is illustrated with experimental tests which validate the proposed approaches and algorithms. Finally, we conclude and present some perspectives in Chapter 7.

Chapter 2

Fundamentals

Since all problems considered in this work stem from the realm of graph theory, we fix in this chapter some elementary graph-related terms and notations. We also introduce various routing problems and shortest path algorithms. Some general terms are also introduced to make readers familiar with single solution and population based metaheuristics.

Sommaire

Related publications:

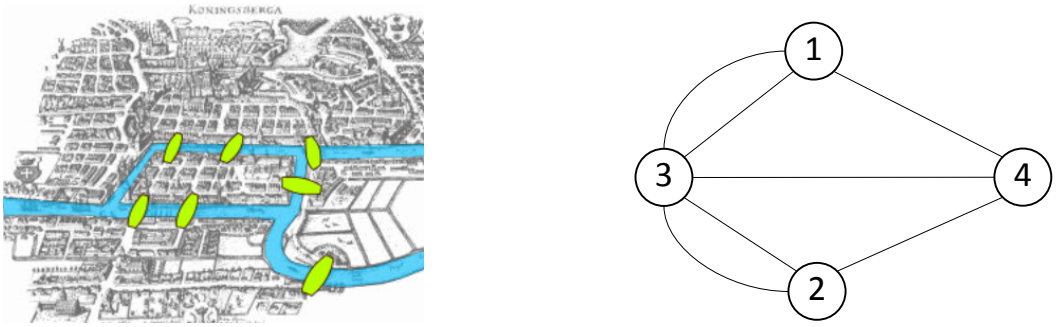
1. Omar Dib, Marie-Ange Manier, Laurent Moalic, and Alexandre Caminada. A Hybrid Metaheuristic for Routing in Road Networks. Intelligent Transportation Systems (ITSC). 2015 IEEE 18th International Conference on Intelligent Trasnportation Systems. pages: 765–770, IEEE, 2015. (Qualis: B1).

2. Omar Dib, Alexandre Caminada, and Marie-Ange Manier. Genetic algorithm combined with variable neighborhood search to solve the one-to-one shortest path problem. 11th Metaheuristics International Conference MIC, June 7-10 Agadir, MIC15, 2015, (Qualis: B3).

2.1 Graphs

In an article published in 1736, Leonhard Euler proved that it is impossible to take a walk through the town of Konigsberg (today Kaliningrad), visiting each part of the town and crossing each of its seven bridges only once [**?**]. A new data structure called "*graph*" was used to prove the theorem.

A graph is a simple data structure that can be used to model problems in a wide range of fields such as telecommunications, planification, electronic, transportation or even in the theory of complexity.



(a) Konigsberg city (b) Graph representation

Figure 2.1 – The city of Konigsberg with its graph representation

Google Scholar references nowadays more than five millions papers containing the word *graph* in their title. Due to the enormous works that have been done in the graph theory area, it is not possible to present every detail in this field. Therefore, we focus in this state of the art on some notations related to routing issues in transportation networks.

A Notations

To put it in a nutshell, a graph defines the existence of relationships between objects such as a segment between two metro stations, a friendship between two people in social media or even a street between two road junctions.

B Nodes and Edges

The objects in a graph are called nodes or vertices and the relationships are called edges or links. Let $\mathcal{N}$ the set of n=$|\mathcal{N}|$ nodes where $||$ refers to the cardinal of a set and $\mathcal{A}$ the set of m=$|\mathcal{A}|$ edges. Since an edge always connects two nodes (but two

nodes are not necessarily connected), we can write $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$. $\mathcal{G}(\mathcal{N}, \mathcal{A})$ defines therefore a graph.

C Graph Orientation

When the relationships between objects are symmetric, the graph is called undirected. Formally, a graph is undirected if $\forall(u, v) \in \mathcal{A}$, $(v, u) \in \mathcal{A}$. A computer network is usually considered as an undirected graph since the communication between two computers is bidirectional [**?**].

In contrast, a road network is usually represented by a directed graph since some roads are unidirectional. Usually, in a directed graph, an edge $a \in \mathcal{A}$ is uniquely identified by its source $u \in \mathcal{N}$ and target $v \in \mathcal{N}$, and we just write (u,v) instead of a. It is noticed from the literature that the terms *node* and *edge* are usually used in directed graphs and *vertex* and *link* are used for undirected graphs.

D Edges' Weights

In many problems, it is desirable to describe the relationships between objects. For this seek, an edge in a graph may be associated with a weight so that it becomes easy to quantify/qualify the relationship [**?**]. For instance, in social media, the weight can define the nature of the relationship (*e.g.* friend, family, colleague). The weight in transport network can represent other metrics such as the distance of a segment or the time required to travel along the segment using a transport mean. Sometimes the terms cost/weight can be used interchangeably.

By associating each edge $\forall a \in \mathcal{A}$ with a cost $c_a \in \mathcal{C}$, the weighted graph $\mathcal{G}(\mathcal{N}, \mathcal{A}, \mathcal{C})$ is obtained. In many cases, the cost of an arc is static, but we will see later in this paper that a deterministic or stochastic function can be used to describe the weight of a segment in the transportation network.

E Successors and Predecessors

The function Γ^+ defines for each node the list of successors, let $\Gamma^+(u) = \{v|(u, v) \in \mathcal{A}\}$. Similarly, we define the predecessors function $\Gamma^-(u) = \{v|(v, u) \in \mathcal{A}\}$. When the graph is symmetric, the predecessors and successors of a node are the same. Generally speaking, two vertices are called *adjacent* if they are linked by an edge, and an edge a is incident with a node n (and vice versa) if n is contained in the ordered pair or unordered set constituting a, respectively.

F Degree and Density

The degree of a node is the number of adjacent edges of the node and the density of a graph is the relation $\frac{m}{n}$ between the number of nodes and edges in $\mathcal{G}$. If we consider the road network of the *Manhattan* city where the streets are always linked to a 4-branches junction, the degree of one node is therefore close to eight.

The density of a graph is an important point to consider since it may highly influence the running time complexity of routing algorithms. While some algorithms efficiently work in sparse graphs, some algorithms may suffer from unacceptable running time in dense networks.

G Graph Representation

There are several data structures that can be used to represent a graph in practice. The most two famous representations are Adjacency Matrix and Adjacency List [**?**]. In the former, it is a two-dimensional matrix, in which the rows represent source nodes and columns represent target nodes. Data on edges and vertices must be stored externally. Only the cost for one edge can be stored between each pair of vertices. In the latter, nodes are stored as records or objects, and every vertex stores a list of adjacent nodes. This data structure allows the storage of additional data on the nodes. Additional data can be stored if edges are also stored as objects, in which case each vertex stores its incident edges and each edge stores its incident vertices.

Choosing which data structure is the most efficient is not an easy question since it highly depends on the structure of the graph itself and the allowed resources (*e.g.* space memory) to use the graph. The graph representation has a direct impact on the complexity of algorithms used to solve routing issues.

H Path

A path (also called itinerary or route in the transportation context) p in G is a sequence of edges $\langle a_1,...,a_k\rangle$ such that the target of a_i and the source of a_{i+1} are the same for all i, $1 \leq i \leq (k-1)$ [**?**]. The first node in a path is usually called the *origin* and the last node is called the *destination*.

In a weighted graph, the length $c(p)$ of a path p can be computed by summing

the weight of all edges included in p (see Equation (2.1)).

$$c(p) = \sum_{i=1}^{l-1} c_{u_i,u_{i+1}} \tag{2.1}$$

l represents the number of edges in p, u_i the i^{th} node in p and u_{i+1} its successor, and c is the cost of the edge (u_i,u_{i+1}).

More advanced approaches should be implemented to evaluate the length of a path having edges with several costs or one cost that is not represented by a single static value.

2.2 Shortest Path Problems

Shortest path problems have received significant attentions from both researchers and practitioners. Several real world problems have been modelled as shortest path problems in directed or undirected graph. In the transport field, several routing issues can be found in the literature. We present in the following some of them, with a special focus on their solving complexity.

A Earliest-Arrival Problem

In railway applications, a query (O, D, t_0) consists of a departure station O, arrival station D, and a departure time t_0 (including the departure day). Connections are *valid* if they depart at least at the given t_0, and the optimization criterion is to minimize the difference between the arrival time and the given departure time [**?**]. We distinguish between two different variants of the problem: (a) The *simplified version*, where train transfers take negligible time and, hence, the input is restricted to transfer = 0 for all stations. (b) The *realistic version*, where train transfers require arbitrary non-negative minimum transfer times. A special case of the earliest-arrival problem is the latest-departure problem: among all connections with earliest-arrival time at the arrival station, maximize the actual departure time at the departure station.

B Minimum Number of Transfers Problem

A query consists only of a departure station O and an arrival station D. Trains are assumed to operate daily, and there is no restriction on the number of days a

timetable is valid. All connections from O to D are valid and the optimization criterion is to minimize the number of train transfers. In particular, let $P = (e_1, ..., e_k)$ be a connection from O to D and let $trans_i(P) \in \{0, 1\}$ be a variable denoting whether a transfer is needed from elementary connection (e_i,$1 \leq i < k$). Then, $trans_i(P) = 0$, if the transport mode $Z(e_{i+1})$ of the edge (e_{i+1}) is the same as the transport mode $Z(e_i)$ of the edge (e_i), and $trans(P) = 1$, otherwise. Consequently, the objective of this routing problem is to minimize, among all P, the quantity $\sum_{i=1}^{k-1} trans_i(P)$ [**?**].

C Resource Constrained Shortest Path Problem

The resource constrained shortest path problem (RCSP) asks for the computation of a least cost path obeying a set of resource constraints. Given a graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$, a source node s and a target t, and k resource limits $\lambda^{(1)}$ to $\lambda^{(k)}$. Each edge e has a cost c_e and uses $r_e^{(i)}$ units of resource (i,$1 \leq i \leq k$). Cost and resource usages are assumed to be non-negative. They are additive along paths. The goal is to find a least cost path from s to t satisfying the resource constraints. CSP has applications in mission planning and operations research. CSP is NP-complete even for the case of one resource [**?**].

D Multiobjective Shortest Path Problem (MOSP)

Computing itineraries with respect to several criteria refers to the Multiobjective Shortest Path problem (MOSP), a fundamental problem in the field of multiobjective optimization [**?**]. Solving this problem consists in finding the set of nondominated journeys from which the user chooses his most preferred one. Given two journeys j_1 and j_2, we say that j_1 dominates j_2 if there is at least one criterion for which j_1 has a better value than j_2 and there is no criterion for which j_2 has a better value than j_1. A journey j is then called Pareto-optimal if it is not dominated by any other journey. The main difficulty in multiobjective contexts stems from the fact that, in many optimization problems, determining the entire set of nondominated solutions is a tedious task since one problem may have a huge number of nondominated solutions (even in case of two objectives).

This problem can be formally described as follows: Given a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of vertices and $\mathcal{A}$ is the set of edges with cardinality $|\mathcal{N}| = n$ and $|\mathcal{A}| = m$ and a *d-dimensional* function $w : \mathcal{A} \rightarrow [R^+]^d$. Each edge e belonging to $\mathcal{A}$ is associated with a weight vector $w(e)$. A source vertex s that represents a departure station, a target vertex z that represents the arrival station and a departure time dt are identified. A path p is a sequence of vertices and edges from s to z with respect to dt [**?**].

The cost vector $W(p)$ for linear functions of path p is the sum of the weight vectors of its edges, that is $W(p) = \sum_{e \in p} c(e)$. Given the two vertices s and z, let $P(s, z)$ denotes the set of all $s - z$ paths in $\mathcal{G}$. If all objectives are to be minimized, a path $p \in P(s, z)$ dominates a path $q \in P(s, z)$ iff $W_i(p) \leq W_i(q)$ for all indices $i, i \in \{1,\ldots,d\}$ and $W_j(p) < W_j(q)$ for at least one index j, $j \in \{1,\ldots,d\}$. A path p is Pareto-optimal if it is not dominated by any other path. The set of all nondominated solutions is called the Pareto-optimal set. The ultimate goal is then to compute the set of nondominated solutions Q ($Q \in P(s,z)$) of $P(s,z)$ with respect to c and the departure time dt.

Other variants of this problem has been introduced such as finding the solution that minimizes one criterion while retaining the second below a given threshold, or finding the lexicographical first Pareto-optimal solution (*e.g.* find among all connections that minimize the earliest arrival, the one with the minimal number of transfers) [**?**].

E Least Expected Shortest Path Problem

Considering a network with $\mathcal{N}$ nodes and various connections between them and giving an origin-destination pair, the main objective is to define the sequence of nodes to be visited such that the lowest expected travel time is achieved. All link travel times are assumed to be positive, and time invariant random variables [**?**]. This problem can be further extended so that link travel times are time dependent and correlated with each other. This correlation can be seen in real world contexts such as in metro network where a delay in one vehicle may cause a delay for the other vehicles.

Other variants of shortest paths can be computed in a stochastic environment such as finding the most reliable shortest path, *i.e.* the one that maximizes the probability of arriving on time or before a predefined upper bound time or within a range of time.

2.3 Shortest Path Algorithms

To solve basic and advanced shortest path problems, several algorithms have been proposed in the literature. While the focus on some methods is to compute the optimal solution(s), others aim at finding near optimal solutions with a guarantee on the running time. We present in the following several shortest path algorithms, which have been developed and adapted to solve a wide range of routing algorithms, either in the transportation area or other fields such as energy, communication networks, social media and artificial intelligence.

A Exact Methods

With an exact algorithm, the optimality of solutions found are always guaranteed. Obviously, developing an exact algorithm for solving an optimization problem is the most preferred option. However, finding the optimal solution(s) often requires a large amount of computational effort. Thus, exact algorithms may not always be the most efficient approaches to deal with real world problems. In the following, we present several exact shortest path algorithms, which have been widely used to solve several variants of shortest path problems.

A.1 Dijkstra

The classical algorithm for computing the shortest path from a given source to all other nodes in a directed graph with non-negative edge weights is due to Dijkstra [**?**]. The algorithm maintains, for each node u, a label $distance[u]$ with the tentative distance from the origin node s to the destination node t. A priority queue Q contains all nodes that depict the current search horizon around s. At each step, the algorithm removes (or settles) the node u from Q with minimum distance to s. Then, all outgoing edges (u,v) of node u are relaxed, *i.e.* we check whether $distance(s,u) + length(u,v) < distance[v]$ holds. If it holds, a shorter path to v via u has been found. Hence, v is either inserted to the priority queue or its priority is decreased.

Algorithm 1: Dijkstra algorithm

input : $(\mathcal{G}(\mathcal{N},\mathcal{A}),s,t)$
output: The path with the minimum distance from s to t

```
1  Q.add(source, 0)                                          // priority queue
2  distance[]                          // array for distances of nodes from source s
3  while !Q.empty() do
4      u ← Q.deleteMin()                                     // settling node u
5      for all outgoing edges e from u do
6          v ← e.head                                        // relaxing e
7          if ((distance[u] + e.weight()) < distance[v]) then
8              distance[v] ← (distance[u] + e.weight())
9              key ← distance[v]                  // key depends on distance to s
10             Enqueue(v, key)                        // insert or decreaseKey
11         end
12     end
13 end
```

The algorithm terminates if all nodes are settled. Algorithm 1 gives the algorithm in pseudo-code. The asymptotic running time of Dijkstra's algorithm depends

on the choice of the priority queue. For general graphs, Fibonacci heaps [**?**] yield a worst-case time complexity of $O(m + n \log n)$, while using binary heaps [**?**] results in a worst case time complexity of $O(m \log n)$

Since transportation networks are normally sparse, a binary heap yields the same asymptotic time complexity in such networks. The advantage of binary over Fibonacci heaps is that the former are easier to implement and yield a lower computational overhead. Moreover, the search horizon of speed-up techniques is normally very low, so the impact of priority queues on the running times fades. In this work, we therefore use a binary heap as priority queue.

If we are only interested in computing a shortest path from s to a given target t, we may stop the search as soon as t is settled by Dijkstra's algorithm. The priority of all nodes settled after t have a greater distance to s and since the graph contains only positive edge weights, the distance label of t cannot be improved further. Running Dijkstra from a given source can be interpreted as growing a tree. If we store the predecessor of each node, we can easily identify the shortest path tree edges.

A.2 A*

Dijkstra's algorithm scans all vertices with distances smaller than $dist(s, t)$. Goal-directed techniques [**?**] such as A* algorithm, in contrast, aim to *guide* the search toward the target by avoiding the scans of vertices that are not in the direction of t. That can be done by exploiting either the geometric embedding of the network or the properties of the graph itself, such as the structure of shortest path trees toward compact regions of the graph.

The basic idea behind A* algorithm is to use a potential function π: $\mathcal{N} \to \mathbb{R}$ on the vertices, which is a lower bound on the distance $dist(u, t)$ from u to t. It then runs a modified version of Dijkstra's algorithm in which the priority of a vertex u is set to $dist(s, u) + \pi(u)$. This causes vertices that are closer to the target t to be scanned earlier during the algorithm.

The algorithm can be made bidirectional, but some care is required to ensure correctness. A standard approach is to ensure that the forward and backward potential functions are consistent. In particular, one can combine two arbitrary feasible functions π_f and π_r into consistent potentials by using $\frac{\pi_f - \pi_r}{2}$ for the forward search and $\frac{\pi_r - \pi_f}{2}$ for the backward search [**?**]. Another approach is to change the stopping criteria instead of making the two functions consistent [**?**], but this is more complicated and performs no better in practice.

In road networks with travel time metric, one can use the geographical distance between u and t divided be the maximum travel speed (that occurs in the network)

as the potential function. Unfortunately, the corresponding bounds are poor, and the performance gain is small or non-existent [**?**]. In practice, the algorithm can be accelerated using more aggressive bounds (for example, a smaller denominator), but correctness is no longer guaranteed. In practice, even when minimizing travel distances on road networks, A* with geographical distance bound performs poorly compared to other modern methods.

One can obtain much better lower bounds (and preserve correctness) with the ALT *(A*, landmarks, and triangle inequality)* algorithm [**?**]. During a preprocessing phase, it picks a small set $\mathcal{L} \subseteq \mathcal{N}$ of *landmarks* and stores the distances between them and all vertices in the graph. During a distance query, it uses triangle inequalities involving the landmarks to compute a valid lower bound on $dist(u, t)$ for any vertex u. More precisely, for any landmark l, both $dist(u, t) \geq dist(u, l) - dist(t, l)$ and $dist(u, t) \geq dist(l, t) - dist(l,u)$ hold. The corresponding potential function is feasible [**?**].

The quality of the lower bounds (and thus query performance) depends on which vertices are chosen as landmarks during preprocessing. On road networks, picking well-spaced landmarks close to the boundary of the graph leads to the best results, with acceptable query times on average [**?**]. For a small (but noticeable) fraction of the queries, however, speedups relative to bidirectional Dijkstra are minor.

A.3 Research Status

Routing is a widely studied topic in transport systems, mainly because of its relevance to real world applications in a wide range of fields such as energy, military and communication networks.

The major research effort on this problem relates to two things: modelling a transport network and solving routing issues. While the former consists of defining how to adequately represent a transport system, the latter deals with developing efficient strategies to support routing issues faced by passengers and transport operators.

In terms of modelling, [**?**], [**?**], [**?**] have done extensive works to incorporate the multimodality aspect into their models. [**?**] proposed a switch point approach to model multimodal transport networks. [**?**] conducted several researches for efficiently designing multimodal transport networks. [**?**] proposed also a transfer graph approach for multimodal transport problems. [**?**] introduced a generic method to construct a multimodal transport network representation by using transfer links, which is inspired by the so-called super-network concept. [**?**] has also done relevant works to generalize a time-expanded model that deals with realistic transfers. [**?**] also handled multimodal networks by incorporating predefined transfer arcs between

nearby stations.

When it comes to routing algorithms, several approaches have been proposed for solving basic and advanced routing problems. For instance, [**?**] adapted the algorithm of Dijkstra to take into account the time dependency and the multimodality aspect of the transport system. [**?**] described an algorithm for itinerary planning based on dynamic programming. [**?**] performed a study on handling times and fares in a routing algorithm for public transport. [**?**] solved the earliest arrival problem on the time-expanded model in a straightforward manner by using a modified version of the algorithm of Dijkstra.

Computing MOSP has been also studied recently. For instance, [**?**] proposed a backward label-setting algorithm for identifying important solutions for the all to one multiple criteria time-dependent shortest path. [**?**] also used a linear utility function that incorporates travel time, ticket cost, and *inconvenience* of transfers. Moreover, other label setting algorithms such as [**?**], [**?**] and label correcting algorithms such as [**?**] have been modified for solving the MOSP.

Although the above-mentioned works on handling multicriteria using straightforward approaches are very significant, however, they have some drawbacks. From one side, they usually handle the multicriteria problem by transforming it into an easier single criterion problem. Thus, the decision maker will surely loose some interesting solutions. From another side, such approaches may cause exponential running time during the resolution phase of the problem. That is, classical approaches may suffer from a high computational time, which make them unusable within real world routing system where passengers seek real time answers.

To overcome such drawbacks, several works have focused on applying heuristic approaches such as metaheuristics to provide high quality solutions within reasonable computational time. Metaheuristics usage is not only limited to the transportation field but it can also be found in various areas such as networking, scheduling and logistics, as well as many optimization problems.

In stochastic networks, several approaches have been proposed in order to solve shortest paths; however, in most of them, the travel time was the one and only one considered criterion. For instance, [**?**] proposed an efficient algorithm to compute the least expected shortest path in a time-varying network. In their work, stochastic edges are transformed into deterministic edges by using the expected value of their distribution functions. Experimentations show that their proposed algorithm succeeded in providing more robust solutions than traditional deterministic approaches; however, that was at the expense of nonpolynomial runtime complexity even when a single criterion is considered.

Other approaches were also proposed for various extensions of shortest path problems in stochastic networks such as in [**?**] where an efficient algorithm is pro-

posed in order to solve the least expected shortest path with having a guarantee to arrive at the destination node before a given time.

B Metaheuristics

Combinatorial optimization is an important branch of operation research. It widely exists in the area of economic management, industrial engineering, information technology, communication networks, etc. Combinatorial optimization problem consists in finding the optimal solutions from all the solutions under a given optimal condition. The form of combinatorial optimization problems is diverse, but its general mathematical model can be described as follow:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \mathrm{f}(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & x \in D \end{aligned} \tag{2.2}$$

where x is the decision variable, $f(x)$ is the objective function, $g(x)$ is the constraint, and D denotes the domain of x. $F = \{x|x \in D, g(x) \leq 0\}$ is the feasible region. Any element in F is a feasible solution of the problem, and F is a finite set. Usually D is also a finite set. Therefore, as long as F is not an empty set, theoretically the optimal solution can be obtained through exhaustive search for the elements of D.

There are many classic combinatorial optimization problems in operations research, such as the traveling salesman problem (TSP), the knapsack problem, the vehicle routing problem (VRP), the scheduling problem, the bandwidth minimization problem (BMP), etc.

Theory shows that these problems are NP-hard problems, so they can only rely on heuristic algorithms to find a local optimum or a feasible solution. Heuristic algorithm is proposed with respect to the exact algorithm. The exact algorithm purpose is to obtain the optimal solution for problems, but its computing time may be unacceptable. Especially in engineering applications, computing time is an important indicator of the algorithm feasibility. Therefore, exact algorithms can only be able to solve comparatively small size problems with a reasonable running time. In order to balance the relationship between calculation costs and the quality of results, heuristic algorithms began to be used to solve combinatorial optimization problems.

The heuristic algorithm is defined in several different descriptions in the literature. It is an intuitive or experienced construction algorithm. In an acceptable

cost (time, space, etc.), a feasible solution for each instance of the combinatorial optimization problem may be given, but the gap between that solution and the optimal solution cannot be considered.

A heuristic algorithm has the following advantages: it is simple, intuitive, and the solving time is fast; some heuristic algorithms can be used in an exact algorithm, such as in the branch and bound algorithm, where they can be used to estimate the upper bound.

Meanwhile, there are some weaknesses of heuristic: it cannot guarantee to obtain the optimal solution; the quality of the algorithm depends on the real problem, the designer's experience and the programming technology.

However, before the end of the 1980s, most of the proposed heuristics for the combinatorial optimization problem were dedicated to a given problem [**?**]. Therefore, more general techniques have been proposed, known as metaheuristic. Because the metaheuristic does not excessively rely on the structure information of the problems, it can be applied to many types of combinatorial optimization problems.

The term metaheuristic was first used by Glover in 1986 [**?**], which derives from two Greek words: heuristic comes from the verb *heuriskein*, which means *to find*, and the prefix meta which means *beyond, in an upper level* [**?**]. Before metaheuristic was widely used, it was often called modern heuristics [**?**].

So far there is no commonly accepted definition for metaheuristic, but there are several representative definitions. Osman and Laporte [**?**] gave the definition: "A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, and learning strategies are used to structure information in order to find efficiently near-optimal solutions".

In summary, metaheuristic is the technique which is more general than heuristic. Metaheuristic can find a sufficiently good solution or even the optimal solution for the optimization problem with less computational assumptions. Therefore, in the past decades, many researches have been focused on using metaheuristics for solving complex optimization problems.

In sections B.1, B.2, and B.3 we introduce a number of well-known metaheuristics in details. Each metaheuristic will be presented with its basic scheme and research status. We also devote on section to discuss two ways to improve the global search capability of the metaheuristic algorithm. Section B.4 introduces three types of evaluation for metaheuristic performance.

B.1 Variable Neighbourhood Search (VNS)

The single solution based metaheuristic (also called trajectory method) has only one solution in the iterative process. The commonality of this kind of metaheuristic is that there is always a mechanism to ensure that the inferior solution could be accepted and become the next state, and not just greedily select the best state.

The core processes of the single solution based metaheuristic contains two steps: select the candidate solutions, determine and accept the candidate solution. In the first step, the generation of candidate solutions is dependent on the solution expression and the selection of the neighbourhood function, but this step is often associated with the structure of optimization problems.

As in the TSP problem, random swap and k-swap are the common methods to generate neighbourhood solution. The second step is the difference among these algorithms. For example, the Tabu Search produces multiple candidate solutions, and deterministically chooses the best state based on the tabu list and the aspiration criterion [**?**], or the Simulated Annealing generates a candidate solution, and accepts inferior solutions with a probability. We detail in the following a third method called Variable Neighbourhood Search [**?**] that we will use in our work.

VNS is a metaheuristic that was firstly proposed by Hansen and Mladenovic [**?**] in 1997. VNS has been proved to be very efficient for obtaining approximate solutions to a wide range of optimization problems. VNS includes dynamically changing the neighbourhood structures. The algorithm is more general when the degree of freedom is large, and many variants can be designed for specific problems.

Basic Scheme
VNS relies on the following three facts [**?**]:

1. The local optimum of a neighbourhood structure is not necessary the local optimum of another neighbourhood structure.

2. The global optimum is a local optimum for all possible neighbourhood structure.

3. For a lot of problems, the local optimums of several neighbourhood structures are close to each other.

The last fact is obtained from the experience, it means that the local optimal solution can provide some information of the global optimal solution. Through the study of the neighbourhood structure, better feasible solutions can be found, and then VNS keeps close to the global optimal solution.

When using neighborhood change to solve the problem, neighborhood transformation can be divided into three categories [**?**]: deterministic, stochastic, and both deterministic and stochastic.

$N_k(k = 1, 2, ..., k_{max})$ is defined as a finite set of neighbourhood structures, where $N_k(x)$ is the solution set of k neighbourhoods for x. The basic procedure of neighbourhood change is to compare the value of the fitness between the new solution x' and the current solution x in k_{th} neighbourhood $N_k(x)$. If the new solution has improved, then $k = 1$ and the current solution is updated ($x \leftarrow x'$). Otherwise, the next neighbourhood will be considered ($k = k + 1$).

Algorithm 2: Pseudo code of Variable Neighbourhood Search (VNS)

input : A set of neighbourhood structures N_k,(k = 1,2,...,k_{max})
A random initial solution x

```
1  while Stopping rule is not satisfied do
2      k = 1
3      while k ≤ k_max do
4          Shaking: One solution x'(x' ∈ N_k(x)) is generated randomly
            from the k_th neighborhood structure of x
5          Local search: Set x' as the current best solution. Do the local
            search in the k_th neighborhood structure N'_k(x') of x', and get
            the local best solution x'' in N'_k(x')
6          Move or Not:
7          if f(x'') < f(x) then
8              x ← x''          // Move to a better solution than the current one
9              k ← 1            // Restart with the first neighborhood structure
10         else
11             k ← k + 1                  // A local minimum has been detected
12         end
13     end
14 end
15 return the best solution
```

Research Status

Hansen and Mladenovic first proposed the Variable Neighbourhood Search algorithm in 1997 [**?**], and then in 2001 they published an improved version of the algorithm [**?**] in which they perform a comparative analysis with classical algorithm for specific optimization problems. In recent years, a large number of papers have been published using VNS as an advanced local search procedure and also several international journals have emerged to deal with research works based on VNS.

Hansen and Mladenovis [?] used VNS and 2-opt algorithm to solve the TSP (the problem size from 100 to 1,000), and the results showed that the VNS obtains an average improvement of 2.73% in solutions quality and average save of 22.09s in running time in comparison with other approaches [?].

[?] designed guided VNS algorithm to solve 32 existing large scale Vehicle Routing Problems (VRP) [?]. The comparison with Tabu Search (TS) [?] showed that the proposed algorithm is better than TS in terms of timelines, and also it solved the VRP problem whose size is up to 20,000 cities. [?] constructed initial solution using saving algorithm, and used 3-opt as the local search strategy. The results showed the effectiveness of VNS compared with previous research.

[?] first introduced VNS to solve the Graph Coloring Problem.[?] adopted VNS to solve this problem, and its neighbourhood design used k-edge exchange method. The experiments showed that VNS is superior to GA in timelines.

In addition, there are many papers using improved VNS to solve combinatorial optimization problems. For example, [?] solved jop shop scheduling problem using VNS combined with GA. [?] solved p-median problem with parallel VNS. [?] presented a hybrid heuristic ordering and VNS for solving the nurse roistering problem. [?] proposed variable neighborhood decomposition search method for 0-1 mixed integer problem. [?] combined VNS and integer linear programming to solve the generalized minimum spanning tree problem. In these problems, VNS have provided good results.

B.2 Genetic Algorithm (GA)

In population based metaheuristics, each generation has multiple individuals with parallel computing. The difference between these metaheuristics is the rule of generating adjacent states (*i.e.* the next state for the population). For example, the Genetic Algorithm makes on certain selected chromosomes evolved with genetic operators and the Scatter Search constructs the subset from the reference set. Genetic Algorithm and Evolutionary Algorithm refer to the same category of algorithms with no real difference nowadays.

Basic Scheme

Genetic Algorithm was proposed by Holland [?] inspired by biological evolution. It is a metaheuristic which is based on the idea of the survival of the fittest individuals. Through the population of individuals evolving generation after generation, while including selection, crossover and mutation operations, the algorithm ultimately converges to the individual which is the best adapted to the environment, and thus obtains either the optimal solution or satisfactory solution.

GA is a general optimization algorithm. The encoding techniques and genetic

operations are relatively simple, and optimization is not restrained by the constraint conditions, so it has a wide range of application. Therefore, GA is widely used in automatic control, computer science, pattern recognition, engineering design, management and social sciences and other fields.

GA is a kind of stochastic optimization algorithm, but it is not a simply random comparison search. Through using the evaluation of individuals and the role worked on chromosome genes, the existing information is effectively used to guide the search which can explore the states and hopefully improves the optimization quality.

The following pseudocode simply illustrates the Genetic Algorithm operation process.

Algorithm 3: Pseudo code of Genetic Algorithm (GA)

```
Initialization: Initialize population
                Calculate the fitness value of initial population
// Iteration
while Stopping rule is not satisfied do
    Selection: According to the fitness value, execute the selection
     operation
    if rand(0,1) ≤ crossover rate then
        Execute the crossover operation                    // Crossover
    end
    if rand(0,1) ≤ mutation rate then
        Execute the mutation operation                     // Mutation
    end
    Replacement: Select individuals to construct the new population
    Update population
    Calculate the fitness value of the new population
end
return the best solution
```

Genetic Algorithm uses the idea of biological evolution and heredity. Different from traditional optimization methods, it has the following characteristics:

1. The problem parameters are encoded as the chromosome. It makes the fitness free of function constraints, such as continuity, conductivity, etc.

2. The search process of GA is starting from a solution set of the problem, not a single individual, so it has an implicit parallel search feature, thus can greatly reduce the possibly of falling into local minimum.

3. All the genetic operation used in GA are random operations. Meanwhile, the search of GA is according to the fitness value information of the individual

without other information.

4. GA has the capability of global search, and its superiority is mainly reflected in:

   (a) It can do the whole space parallel search, and focuses the search on the high performance parts, which can improve the efficiency and avoid local optimum.

   (b) It has inherent parallelism through the genetic operation on the population, it can handle a large number of states, and profit from parallel implementation.

Research Status

Genetic Algorithm provides a common frame for solving complex optimization problems. GAs are nowadays used in a wide range of applications such as transportation, communication networks and energy. With the increasing scale of the problem, the search space of combinatorial optimization problems have expanded dramatically. Sometimes on the current computer enumeration method it is difficult or even impossible to determine their exact optimal solution. For such complex problems, the research should focus on finding satisfactory solutions, and Genetic Algorithm is one of the best tools which seek such satisfactory solutions. Practice has proved that the GAs have been successfully applied to solve NP-hard problem such as traveling salesman problem [**?**], knapsack problem[**?**], bin packing [**?**], layout optimization [**?**], bandwidth minimization problem [**?**], etc.

For instance, [**?**] worked with evolutionary algorithms to compute single source shortest paths using single-objective fitness. [**?**] also proposed a novel GA to find shortest paths in computer networks. [**?**] also worked with GAs to find shortest paths in data networks. [**?**] proposed a GA with a part of an arterial road regarded as a virus to select route to a given destination on an actual map under a static environment. Moreover, [**?**] presented a GA based strategy to find the shortest path in a dynamic network, which adapted to the changing network information by rerouting during the course of its execution. [**?**] proposed a GA based algorithm with a novel fitness function for simultaneous multiple routes searching for car navigation to avoid overlap. In the work of [**?**], a GA was introduced for determining the weights of different criteria, which eventually achieve a series value of each criterion and sum them up as the final cost. [**?**] used GA for Pareto optimal route set searching in order to reduce the number of routes selection criteria. The GA based solution for multimodal shortest path problem presented by [**?**] showed the robustness of this approach through empirical studies and concluded that GA based approaches can efficiently explore the search space in order to find very good multimodal paths.

Other advanced algorithms have been proposed such as in [?]. In the latter's work, the so-called NSGAII (Nondominated Sorting Genetic algorithm II) has been introduced. The main motivation behind introducing this algorithm is to alleviate the difficulties to use nondominating sorting and sharing: the high computational complexity of nondominated sorting, the lack of elitism, and the needs for specifying the sharing parameter σ. To overcome such drawbacks, NSGAII integrated a fast-nondominated sorting approach, a fast crowded distance estimation procedure, and a simple crowded comparison operator. Experimental results have shown that NSGAII outperforms most of traditional approaches when solving several kinds of optimization problems.

Additionally, Genetic Algorithms have been applied on several aspects including mobile robot path planning [?], robot inverse kinematics [?] etc. Indeed, it is difficult to accurately model the artificial systems. Since the origin of the Genetic Algorithm stems from the study of artificial adaptive system, certainly robotics becomes an important application field of them.

Image processing is also an important research field of computer vision where GAs have been applied in pattern recognition [?], image restoration [?], image feature extraction [?], etc. As well, many data mining problems can be seen as an optimization problem. The database can be seen as the search space and mining algorithms can be seen as the search strategy. Applying Genetic Algorithm can then be very usefful and efficient to extract features, hidden knowledge and important rules from these data [?], [?].

B.3 Memetic Algorithm (MA)

Memetic Algorithms (MAs) are a class of stochastic search heuristics in which evolutionary approaches are combined with problem-specific solvers. The later might be implemented as local search heuristics techniques, approximation algorithms or, sometime, even (partial) exact methods.

Basic Scheme

The hybridization in Memetic Algorithms is meant to either accelerate the discovery of good solutions, for which evolution alone would take too long to discover, or to reach solutions that would otherwise be unreachable by a local method alone. As the large majority of Memetic Algorithms use heuristic local searches rather than approximation or exact methods, in what follows we will focus only on the local search added to the evolutionary process. It is assumed that the evolutionary search provides a wide exploration of the search space while the local search can somehow zoom-in on the basin of attraction of promising solutions. MAs have been proved very successful across a wide range of domains such as combinatorial optimization [?], optimization of non-stationary functions [?], multi-objective optimization [?],

bioinformatics [**?**], etc.

Algorithm 4: Pseudo code of Memetic Algorithm (MA)

Initialization: Initialize population

while *Stopping rule is not satisfied* **do**
  Evaluation: Evaluate the fitness value of individuals
  Selection: Select individuals for reproduction
  Crossover: Recombine parents to produce offsprings
  Mutation: Mutate offsprings
  Improvement: Improve offsprings via local search
  Replacement: Select individuals to construct the new population
  Update population
end
return the best solution

Memetic Algorithms have received various names throughout the literature and scientists not always agree what is and what is not MA due to the large variety of implementations available. Some of the alternative names used for this search framework are hybrid GAs, Baldwinian EAs, Lamarckian EAs, Genetic Local Search algorithms, and other names. [**?**] coined the name Memetic Algorithm to cover a wide range of techniques where evolutionary-based search is augmented by the addition of one or more phases of local search. We show in Algorithm 4 the pseudo code of a Memetic Algorithm.

Research Status

Research works have shown in recent years that Memetic Algorithms provide one of the most effective and flexible metaheuristic approaches for tackling hard optimization problems. Several works have been conducted to address the difficulty of developing high-performance universal heuristics by encouraging the exploitation of multiple heuristics.

MAs have been extensively applied for solving combinatorial optimization problems with respect to one or several criteria.

Indeed, several researches have focused in recent years on combining several metaheuristics to solve one single optimization problem. The main motivation for the hybridization process is to achieve better performance by exploiting and combining advantages of the individual pure strategies [**?**].

For instance, in [**?**] the Multi Trip Vehicle Routing Problem is considered. The total travel time is the only considered criterion and the time and the capacity are considered in the form of constraints. A hybrid Genetic Algorithm is proposed. More precisely, a new local search operator based on the combination of standard Vehicle Routing Problem moves and swaps between trips is introduced. The pro-

cedure has been compared with those in the literature and results have shown that MA outperforms previous algorithms with respect to average solution quality. Moreover, a new feasible solution and many best known solutions are found.

[**?**] proposes a fast Memetic Algorithm with multi-level learning strategies for community detection by optimizing modularity. The proposed algorithm adopts a Genetic Algorithm to optimize a population of solutions and uses the proposed multi-level learning strategies to accelerate the optimization process. The multi-level learning strategies are devised based on the potential knowledge of the node, community and partition structures of networks. Extensive experiments on both benchmarks and real-world networks demonstrate that compared with the state-of-the-art community detection algorithms, the proposed algorithm has effective performance on discovering the community structure of networks.

Additionnaly, [**?**] proposes a memetic approach for solving the mutiobjective flexible job shop scheduling problem (MO-FJSP) with respect to the minimization of the makespan, the total workload and the critical workload. The problem is addressed in a Pareto manner, which aims to search for a set of Pareto optimal solutions. The algorithm consists of combining the NSGA-II with a novel local search algorithm. Experimental studies indicate that MAs perform much better than all other existing algorithms.

Moreover, [**?**] proposes a multiobjective Memetic Algorithm to solve the global path planning of wheeled robots. Several criteria are considered such as the path length, the smoothness, and the safety. The algorithm is implemented from a conventional multi-objective Genetic Algorithms with elitist non-dominated sorting and decomposition strategies respectively. Experimental results on both simulated and real environments show that the proposed algorithm is efficient in planning a set of valid tradeoff paths in complex environments.

B.4 Evaluation of Metaheuristics

The performance evaluation index of metaheuristic are mainly of three types: optimizing performance (effect), time performance (efficiency) and robustness. Metaheuristic is usually more concerned about the first two indexes, namely whether the algorithm is able to improve the optimality or the running time [**?**].

1. Effect

   The optimizing performance index includes the error or the deviation between the optimal value of the problem and the best value obtained using an algorithm.

This (Gap) is defined as:

$$Gap = (\frac{v_{opt} - v_{best}}{v_{opt}}) \times 100\% \tag{2.3}$$

where v_{opt} is the optimal value of the problem, and v_{best} is the best value that the algorithm finds. Smaller Gap means better solutions.

There are also many studies directly using the best value as the index of performance evaluation, which is usually used to compare the performance of multiple algorithms, and this method is more convenient.

2. Efficiency

   The time performance can either check the CPU time or directly examine the number of iterations. Time-consuming calculation is as follows:

$$E = (\frac{I_\alpha T_0}{I_{max}}) \times 100\% \tag{2.4}$$

   Where I_α is the average number of algorithm iterations after running several times over, T_0 is the average computational time for one step iteration, and I_{max} is the given maximal number of iterations. The smaller E is, the better the algorithm converges.

3. Robustness

   Robustness is used to measure the algorithm dependence on the initial input and controllable parameters. Robust index can directly use the mean square error of several experiment results, or use the following formula:

$$R = (\frac{v_{opt} - v_\alpha}{v_{opt}}) \times 100\% \tag{2.5}$$

   where v_α is the average value of algorithm after several executions, and v_{opt} is the optimal value of the problem. The smaller R is, the better the robustness is.

2.4 Synthesis

In this chapter, we have presented general terms related to graph theory and routing problems. We have also outlined several routing algorithms that are nowadays used

to answer passenger's routing requests. The advantageous and limitations of traditional shortest path problems have been also discussed. Metaheuristics and mainly Genetic Algorithms, Variable Neighbourhood Search and Memetic Algorithms have been introduced. The basic scheme, flowchart, advantages, limitations and usages of each method have been explained.

It has been also mentioned that using standard modelling approaches for representing the transport system may suffer from several drawbacks such as the capacity of handling complex routing issues, computing mono/multicriteria shortest paths and dealing with the time variant and stochastic nature of the transport system. Therefore, proposing a new representation of the transport is primal to built advanced traveller information systems.

Besides, we have shown that standard routing algorithms may suffer from unacceptable running time when solving complex routing problems in large scale transport networks. Moreover, such algorithms may not be capable of handling the stochastic aspect of the transport system. Therefore, there is a real need to propose more efficient routing algorithms to handle the various constraints of both passengers and transport operators.

Chapter 3

Network Modelling

This chapter is devoted to present the modelling approach that has been developed in order to represent a multimodal transport network. Both individual transport modes that offer continuous services such as car and bike, and collective modes that work according to predefined timetables such as railway and tram are taken into consideration. Each transport mode is represented as a separate directed graph, and an additional effort is then done in order to merge all sub networks into one larger multimodal network. Combining subnetworks and modelling the general cost of an edge are tackled in Section 3.3. Finally, we introduce in Section 3.4 a general mathematical formulation based on the proposed modelling approach in order to formally describe various routing issues.

Sommaire

Related publications:

1. Omar Dib, Marie-Ange Manier, Laurent Moalic, and Alexandre Caminada. A multimodal transport network model and efficient algorithms for building advanced traveler information systems. Transportation Research Procedia. Elsevier, 22: 134–143, 2017.

2. Omar Dib, Marie-Ange Manier, and Laurent Moalic. Advanced modeling approach for computing multicriteria shortest paths in multimodal transportation networks. Intelligent Transportation Engineering (ICITE), IEEE International Conference on Intelligent Trasnportation Engineering, pages 40–44, IEEE, 2016.

3.1 Individual Transport Modes

A Pedestrian Network

The construction of the model related to the foot network is quite straightforward. Junctions are represented by nodes in a directed graph $\mathcal{G}$. An edge $e(u, v)$ is inserted between u and v if u is linked with v by a footpath [**?**]. Arc costs depend on geographical length and on average pedestrian walking speed. Labels on arcs mark the type of path, such as side-way, stairs, trail, etc. (see Figure 3.1 on page 42 for an example). Nodes can also be associated with labels in order to account for location of interests near nodes.

B Bike Network

The bicycle network is constructed in a similar way to the foot network. Nodes represent road junctions and arcs are inserted into the graph whenever biking is allowed between two junctions.

Different road types for cycling exist, such as roads with a separate cycling path, roads with a separate lane for cyclists which are shared with buses, roads shared with general traffic, etc. Arcs may be labelled accordingly. Arc costs represent cycling time and depend on geographical length and on average cycling speed. Costs may also depend on the time at which the segment is travelled.

The bike network can be used by passengers who want to accomplish their travels using their own bikes. Other services can also be proposed to exploit this network such as bike rental. Indeed, in recent years, most of modern cities have implemented bicycle rental systems (*e.g.* Vélib in Paris (France), Santander in London (UK)). It is obvious that providing affordable access to bicycles for short-distance trips can constitute a very important alternative for a wide range of users. These latter may prefer using a bike to taking a subway. Bike rental systems may also be useful to cover the last kilometre, *i.e.* the distance between a bus or railway station and the departure or final location. For all those reasons, integrating the possibility to rent a bike into a multimodal representation becomes essential.

In this thesis, we consider bike sharing systems with fixed rental stations. In such systems, a set of bike rental stations is distributed over the served areas; users can then retrieve and return bicycles according to their needs and preferences.

Each rental station r is characterized by a maximal number of bicycles it can hold $c_r^{max} \in Z^+$. The number of bikes that can be rent and the free places are also important information to store. By doing so, users are provided with real time information about the state of bike rental stations. Such dynamic information

should be always updated to keep providing users with correct information. A better approach is to provide estimations about the states of rental stations; this is usually done by statistical analysis and observations. The cost of renting a bicycle is also an important information to be provided to users.

The modelling of bike rental stations is straightforward. Special nodes to represent them are inserted in the bike network. These nodes hold information about the state of the rental stations (*i.e.* current/maximal capacity, cost, etc). To account for multimodality, each node is linked to the nearest node in the bike network. This is done by computing a shortest path problem in the bike network. The information in the inserted node are time dependent. An external module will be in charge of updating data. An external prediction module can also be used for providing information about the next states of rental stations.

We show in Figure 3.1 the directed graph used for representing a bike network associated with its bike rental stations.

C Road Network

For the road network, nodes represent road junctions and arcs represent roads connecting the junctions. An edge $e = (u, v)$ is inserted between two junctions $u, v \in \mathcal{V}$ if and only if a road segment from u to v exists in the road network [**?**]. Note that if in reality the road between u and v is a two-way road, then two edges (u, v) and (v, u) are inserted into the final graph.

Different road types exist, *e.g.* local roads, urban roads, inter-urban roads, motorways, toll roads, etc. Arc costs may be time-dependent and represent travel times which depend on the geographical length, the speed limits, and the traffic conditions.

The road network can be used by people who want to drive their own cars while travelling from one place to another. In this case, private or public parkings have to be included in the modelling approach. To do so, special nodes are inserted in the final road network graph. Each node is then linked to the nearest node in the road network to account for transferring from the road network itself to the parking. The state of parking should be periodically updated to always provide users with real time and correct information.

D Car Sharing Network

In recent years, several governments and transport operators have established car rental systems whereby users can rent or share a car to accomplish their journeys.

Electric car sharing such as Autolib in Paris is one example of such systems. A set of electric vehicles is maintained for public use on a paid subscription basis, employing a citywide network of parkings and charging stations.

In this thesis, we propose to model such advanced systems. Indeed, an electric vehicle can share the same network as a normal car. A key difference to road network is that electric car network employs rental stations. Thus, the same idea of bike rental network can be used for representing the electric car network. That is, a special node is inserted for every rental station and it is connected to the nearest node in the road network. Information about the state of rental stations should be periodically updated. For instance, the number of available vehicles should be known at every moment. Such information is primal when proposing itineraries including this transportation mode.

We show in Figure 3.1 the directed graph used for representing a car sharing network. Nodes in green represent car rental stations.

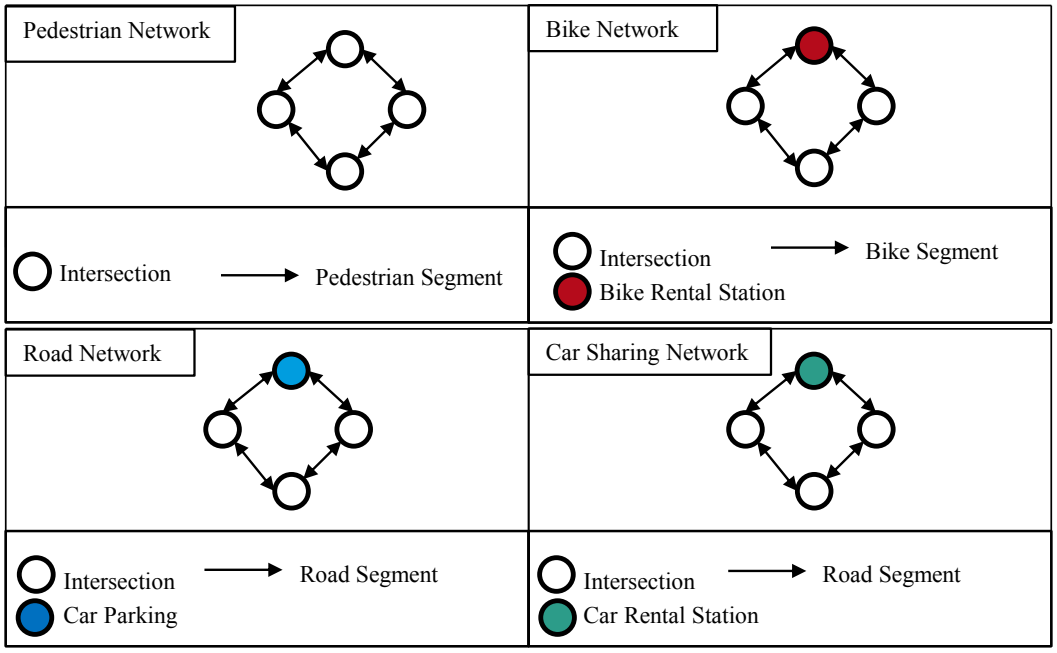


Figure 3.1 – Modelling private transport networks

3.2 Collective Transport Modes

While modelling individual transport modes is quite canonical, representing collective transport modes turns out to be more difficult. A key difference is that collective transit networks are inherently time-dependent, since certain segments of

the network can only be traversed at specific, discrete points in time. As such, the first challenge concerns modelling the timetable appropriately in order to enable the computation of journeys.

Roughly speaking, a timetable (Π, S, R, T, F) where Π is the period of operation, consists of a set of stops S (such as bus stops or train platforms), a set of routes R (such as bus or train lines), a set of trips T and a set of footpaths F (transfers) [**?**]. Trips correspond to individual vehicles that visit stops along a certain route at a specific time of the day. Trips can be further subdivided into sequences of elementary connections, each given as a pair of (origin/destination) stops and (departure/arrival) times between which the vehicle travels without stopping.

As a first step of modelling, we introduce four types of vertices that correspond to stations, access points where people enter or exit a station, platforms and departure events. As in real life, a station may have one or several entrance areas. Therefore, a two-direction edge is inserted between each station node and its entrance points. Note that the weight of such edges is set to 0.

Indeed, several models in the literature ignore the presence of several entrance points for one station; however, in many big cities such as in Paris, taking the wrong exit or entrance in a big station would highly affect the length of an itinerary (*e.g.* the railway station of Chatelet in Paris). Therefore, we decided to model stations' access points to provide more accurate itineraries.

Besides, a station comprises a set of platforms where passengers wait for vehicles. An edge $e(p,ap)$ is inserted between a platform p and an access point node ap if and only if there is a foot path inside the station between p and ap. The weight of $e(p,ap)$ represents the time required for accessing p from ap.

Platforms inside stations are also related via transfer edges to account for the fact that passengers may transfer from one platform to another. The weight of transfer edges inside stations represents the time required to go from one platform to another. As can be noticed, variable transfer times may be associated with one station depending on the physical structures of its platforms. Besides, considering two separate entities for platforms and stations allows more accurate itineraries and more flexibility to handle advanced routing problems such as evacuation.

Furthermore, we assign each platform a type (bus, railway, tram...) to differentiate between modes. This information can be used by routing algorithms to support routing queries that privilege one mode over another. For instance, many passengers would prefer only using the bus along their journeys.

A platform node stores also additional information related to the current load of passengers, who are waiting at the platform. More precisely, the number of passengers at each platform can be stored within the entity itself. The maximum

capacity of a platform is also stored. Such information is primal and plays a very important role when dealing with rerouting issues, as well as, evacuation situations.

Lastly, a platform node is also associated with information related to the station such as the presence of escalators or other services for people with reduced mobility.

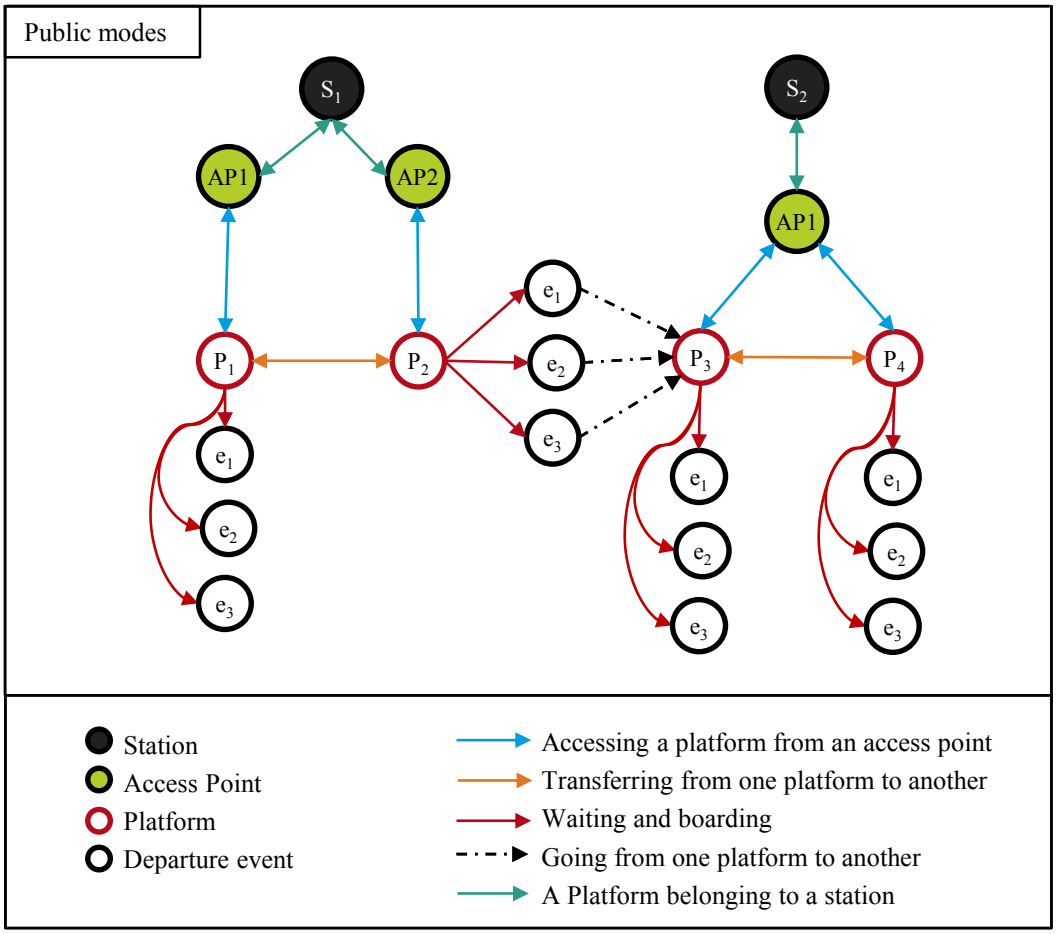


Figure 3.2 – Public transportation network

Since a timetable consists of time-dependent events that happen at discrete points in time, we use the idea of building a time dependent graph to unroll time. Roughly speaking, the model creates a vertex for every event in the timetable that consists of vehicle leaving a platform P_1 at a departure time dt and arrives to another platform P_2 at an arrival time at. Timestamps are inserted into event nodes to account for the departure and arrival times. Event nodes are ordered in the way that a higher-level node refers to an earlier event. This ordering is very helpful to reduce the search space of routing algorithms.

In addition, waiting, boarding and alighting edges are inserted between event

nodes and platforms. More precisely, the edge linking a platform P and an event node e represents the waiting time at the platform and the minimal time required to board a vehicle. Similarly, the edge between an event node e and a platform P represents the minimal time required for alighting from a vehicle to access P.

An event node can store information about the maximum and current capacity of its related vehicle. This information is crucial when dealing with situations that require rerouting of passengers from one vehicle/mode to another with respect to vehicles' capacities. It is worth to mention that by doing so, the capacity of vehicles is considered as a dynamic element, which changes as a function of time.

We show in Figure 3.2 the directed graph related to modelling a collective transport mode. Information stored in each component are presented in Chapter 7.2.

3.3 Multimodal Transport Network

To compute multimodal paths, the routing algorithm has to use multiple networks simultaneously. For that reason, after modelling each network as a separate digraph, the different sub networks have to be combined into one larger multimodal network.

To do so, we use the pedestrian network to transfer from one mode to another. More precisely, the pedestrian network can be seen from two points of view: i) One refers to its usage as a separate mode where passengers can walk to reach their destinations; ii) The second refers to its usage as an intermediate network that passengers can use to transfer from one mode to another.

A Combining Networks

To form a general multimodal graph, each subgraph has to be connected to the subgraph corresponding to the pedestrian network [**?**]. To link the road network, we use nodes corresponding to car parkings. We link each node representing a parking (or rental station) to the nearest neighbour node in the pedestrian network. Therefore, to go from the road network to another network, the passenger has to pass through a parking node.

Linking the bike network is also done through bike rental stations nodes. Each rental station node is linked through an edge to the nearest neighbour node in the pedestrian network. Finally, linking collective networks is done using access points. We link each access point node to the nearest node in the pedestrian network.

It is worth mentioning that finding the nearest neighbour node is done with respect to the Euclidean distance, since car and bike parkings, as well as access

points and pedestrian junctions have longitude and latitude properties. Note also that, the weight of transfer edges is the time required to go from a node to the nearest node in the pedestrian network; this is computed by dividing the distance by the average walking speed.

We show in Figure 3.3 an example of a multimodal graph that contains both individual and collective transport modes, which have been linked using the pedestrian network. As can be seen, passengers can either use the pedestrian network to go from one transport to another, or use transfer arcs from one platform to another in a collective transport mode.

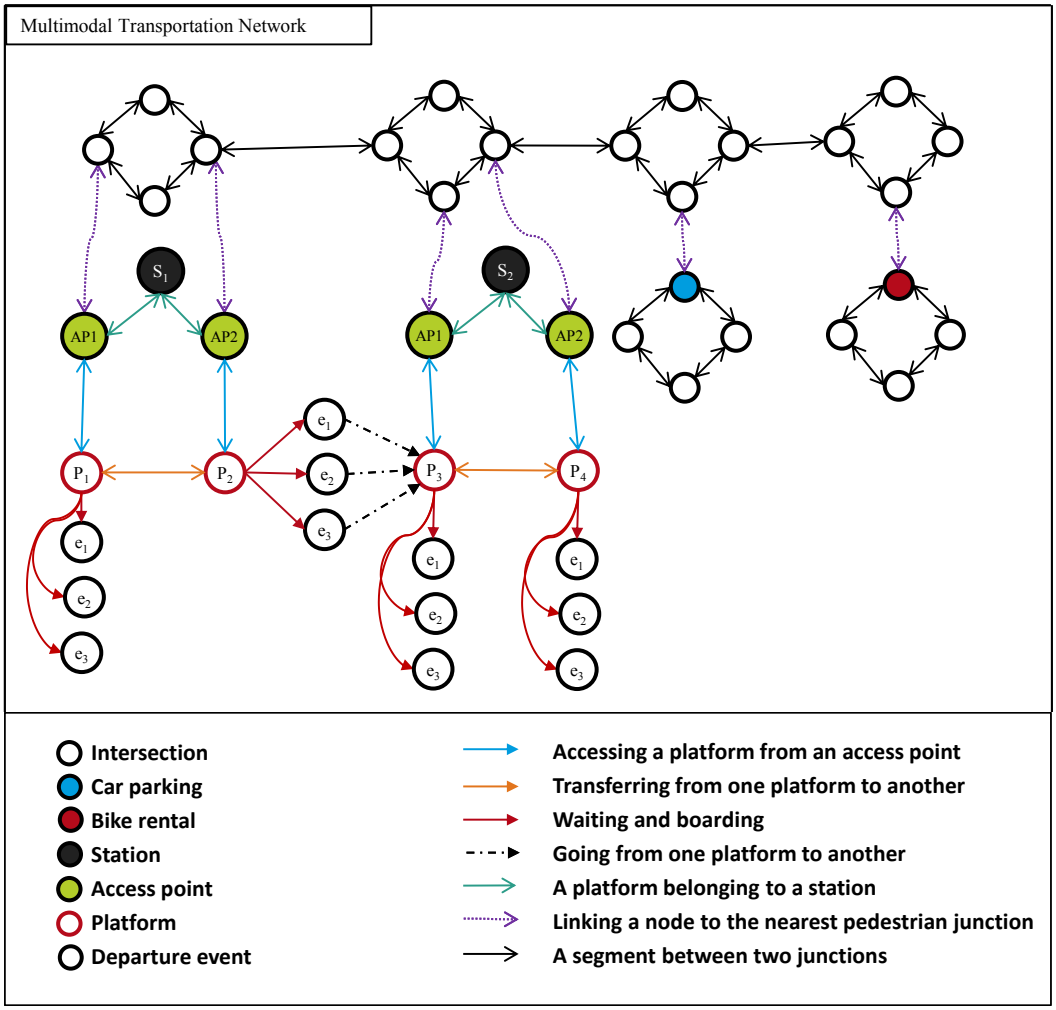


Figure 3.3 – Multimodal network

B Cost Modelling

Edges in this modelling approach can be time independent or time dependent. In the former, the weight of an edge does not depend on the time at which the edge is crossed. However, in the latter category, an edge may have more weight depending on the time the edge is crossed. Furthermore, the weight of an edge may be deterministic or stochastic. In the former, a weight will be represented as a single static value. However, in the latter, the value of an edge's weight comes in the form of a random variable that varies within certain range. We present in the next table the different categories of edges with an example of their travel time.

Table 3.1 – Edges' cost modelling

	Weight	
Edge	**Deterministic**	**Stochastic**
Time-independent	$\omega(e) = \alpha$ min *e.g.* $\omega(e) = 3$ min	$\omega(e) \in [\alpha, \beta]$ *e.g.* $\omega(e) \in$ [1,4] min
Time-dependent	$\omega(e,\varphi) = \alpha$ min *e.g.* $\omega(e,5:00) = 5$ min $\omega(e,7:00) = 10$ min $\omega(e,12:00) = 15$ min	$\omega(e,\varphi) \in [\alpha, \beta]$ *e.g.* $\omega(e,5:00) \in$ [1,4] min $\omega(e,7:00) \in$ [5,15] min $\omega(e,12:00) \in$ [10,25] min

In the stochastic category, the random variable representing the travel time along the edge may vary according to a predefined probability law. It also may vary based on some observations, which leads to a custom probability distribution.

Up to this modelling level, the cost of an edge is always associated with the travel time. However, an edge may be also assigned values to account for other criteria such as the cost of CO2 emission. The presence of such criteria allows later solving multiple criteria routing problems. Other criteria such as the number of transfers, walking time can be assigned on the fly while the routing algorithm accomplishes its search process.

It can be easily noticed that the travel time metric can be always dealt with as a random variable. No one can know with 100 % the exact value of the travel time between two places. For instance, the transfer time between two platforms can be prone to variations due to some unanticipated phenomena (*e.g.* encountering a friend, receiving a call $\cdots$). The departure time of a vehicle can also be regarded as a stochastic element since no one can know the exact value of waiting time at the platforms. Several parameters may affect the travel time such as the behaviours of passengers, drivers or even the vehicles themselves.

Since all edges in the proposed approach are associated with travel time, it is

thus very intuitive to consider randomness among all components in the model. However, this will definitely require a huge amount of time to build the model and also causes a huge amount of running time when solving routing issues. As a result, an efficient way to overcome that is to classify edges with respect to their level of randomness. That is, the randomness is more present in some edges than other.

For instance, the travel time that accounts for a vehicle going from one platform to another is not affected by a lot of parameters. Only, technical problems inside tracks and drivers' behaviours can affect that travel time. As a result, that stochastic variable can be reduced to a static one with fixed travel time value.

However, when it comes to the departure of vehicles, it is crucial to consider the stochasticity at this stage. Many parameters affect the departure of a vehicle such as the number of passengers inside the vehicle and the current load at platforms in addition to the behaviours of passengers and drivers. What we want to say aforementioned, is that some assumptions can be made to consider stochasticity among few edges in the model rather than all edges.

In multimodal networks, advanced mechanisms and theories should be established in order to adequately describe the variation of randomness of the stochastic travel time metric. Such a description is usually translated by analysing a large amount of data related to the random variable itself. For instance, a measurement of the number of passengers can be done over several years in the goal of obtaining enough information to a priori, predict the variation of a stochastic variable. Other approaches are based on simulations, where the whole transport network is emulated. Final states of the model's components can be then obtained with respect to an initial state. Recent works, mainly in machine learning and big data, provide also very important tools to solve prediction issues in transportation and other fields.

Regardless the method used for prediction purposes, it is essential that some aspects have to be respected such as the running time dimension and the quality of predictions. Usually, a compromise has to be made between the quality and time complexity. To measure the quality of predictions, a comparison is usually made between the real travel times (*i.e.* obtained a posteriori) and the predictions themselves computed a priori.

As a result of prediction approaches, travel time distributions should be obtained for every edge in the model. Since edges are time dependent, distributions will also depend on time. We will assume later in this manuscript that travel times vary according to normal distribution law. For instance, the theoretical timetable information indicates that the transfer time between two platforms is α minutes; we will assume that the distribution of this time varies according to normal probability law with average of α minutes and predefined standard variations and minimal/maximal bounds. These information are not static in the proposed approach. They are

only used for implementation purposes. Other probability laws such as exponential, min-max or even custom laws can also be used.

3.4 Mathematical Formulation

After graphically representing a multimodal transport network, we present in the following a general mathematical formulation of various routing problems studied in this thesis. As previously mentioned, in this work, we exploit the proposed modelling approach to solve the shortest path problem with respect to one or several criteria in a time dependent stochastic multimodal transportation network.

The transport system that we are dealing with may be represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{X}, \mathcal{T})$ where: $\mathcal{N}$ is the set of nodes; $\mathcal{A}$ represents the set of arcs; $\mathcal{X}$ defines all transportation modes considered in the network; $\mathcal{T}$ represents the departure time of vehicles during the day.

The set of transport modes available at a node $n_i \in \mathcal{N}$ in order to go to the set of its successors $\Gamma^+(n_i)$ is defined by $\mathcal{X}(n_i) = \bigcup_{n_j \in \Gamma^+(n_i)} \mathcal{X}^{n_j}(n_i)$, where $\mathcal{X}^{n_j}(n_i) = \{x_1^{n_j}(n_i),...,x_k^{n_j}(n_i)\}$; $k = |\mathcal{X}^{n_j}(n_i)|$ and each element $x_1^{n_j}(n_i)$ represents a different transport mode to go from n_i to n_j.

The departure times associated with each transportation mode $x_1^{n_j}(n_i)$ are represented by $\mathcal{T}[x_1^{n_j}(n_i)] = \{t_1[x_1^{n_j}(n_i)],...,t_m[x_1^{n_j}(n_i)]\}$, where m is the number of departures of this transport mode. The set of all departures associated with all transport modes linking one node $n_i \in \mathcal{N}$ with one of its successors $n_j \in \Gamma^+(n_i)$ is defined by $\mathcal{T}[\mathcal{X}^{n_j}(n_i)] = \bigcup_{l=1}^{k} \mathcal{T}[x_1^{n_j}(n_i)]$. Similarly, all departures that link the node $n_i \in \mathcal{N}$ with its successors are represented by the set $\mathcal{T}^{\mathcal{X}(n_i)} = \bigcup_{n_j \in \Gamma^+(n_i)} \mathcal{T}[\mathcal{X}^{n_j}(n_i)]$. Consequently, the set $\mathcal{T}$ of departures associated with transportation modes can be determined as follows: $\mathcal{T} = \bigcup_{n_i \in \mathcal{N}} \mathcal{T}^{\mathcal{X}(n_i)}$.

The parameters related to the travel time and monetary cost associated with an edge along the graph are defined as follows: $\varphi_{n_i n_j}(x,t)$ and $C_{n_i n_j}(x,t)$. These two parameters define the travel time and monetary cost required to go from the node n_i to the node n_j with the transport mode x with respect to the departure time t from n_i.

To account for the dynamic aspect of the transport system, both $\varphi_{n_i n_j}(x,t)$ and $C_{n_i n_j}(x,t)$ can be determined for each couple (x,t) with $x \in \mathcal{X}^{n_j}(n_i)$ and $t \in \mathcal{T}[x]$.

Solving a shortest path problem in a multimodal transport network is performed:

1. in response to a routing request that specifies the input data of the resolution algorithm,

2. using the data related to the transport network,
3. for instantiating the variables, and a number of intermediate variables, so as to minimize a specific objective function,
4. while respecting the set of constraints related to the problem itself.

Data related to the routing query are:

- the source point S;
- the destination point D;
- the departure time dt at which the journey should start from S;
- the set of criteria to be optimized;

Data related to the transport network are:

- the available transportation modes: $\mathcal{X} = \bigcup_{n_j \in \mathcal{N}} \mathcal{X}(n_i)$;
- the departure times at each node: $\mathcal{T}^{\mathcal{X}(n_i)}$, with $n_i \in \mathcal{N}$;
- the travel time along edges $(\varphi_{n_i n_j}(x_i,t_i))$, with $n_i \in \mathcal{N}, n_j \in \Gamma^+(n_i), x_i \in \mathcal{X}(n_i)$, and $t_i \in \mathcal{T}^{\mathcal{X}(n_i)}$;
- the monetary costs along edges $C_{n_i n_j}(x_i,t_i)$ with $n_i \in \mathcal{N}$, $n_j \in \Gamma^+(n_i)$, $x_i \in \mathcal{X}(n_i)$, and $t_i \in \mathcal{T}^{\mathcal{X}(n_i)}$;

Building a solution for the routing problem refers then to instantiate the variables n_i, x_i, t_i, with $n_i \in \mathcal{N}$, $x_i \in \mathcal{X}(n_i)$, and $t_i \in \mathcal{T}[x_i]$.

A route can be formally represented as follows:

$$route = \{(n_1 = S,x_1,t_1),(n_2,x_2,t_2),...,(n_f = D)\} \tag{3.1}$$

where $n_i \in \mathcal{N}$, $x_i \in \mathcal{X}^{n_{i+1}}(n_i)$, $t_i \in \mathcal{T}[x_i]$, $n_{i+1} \in \Gamma^+(n_i)$.

When the value of these variables is set, data related to $(\varphi_{n_i n_{i+1}}(x_i,t_i)$ and $C_{n_i n_{i+1}}(x_i,t_i))$ can be easily retrieved.

The procedure for instantiating these variables is based on the definition and evaluation of a number of intermediate variables which are:

- the earliest arrival time at nodes $n_i \in \mathcal{N} : r_{n_i}$;

– the number of transfers used up to each node n_i : $\mathcal{T}r(n_i)$;

– the walking time up to each node n_i : $\mathcal{W}r(n_i)$;

– the waiting time at each node n_i : $\Psi^{x_{i-1}x_i}_{n_{i-1}n_i ni+1}(t_{i-1},t_i)$.

Determining the value of each of these variables at each node n_i is performed as follows:

– the earliest arrival time is equal to the sum of the departure time t_{i-1} of the predecessor n_{i-1} and the value of the travel time associated with the edge (n_{i-1}, n_i) using the mode x_{i-1}:

$$r_{n_i} = t_{i-1} + \varphi_{n_{i-1}n_i}(x_{i-1},t_{i-1}) \tag{3.2}$$

– the number of transfers is calculated from:

$$\begin{aligned} &\mathcal{T}r(n_i) = \mathcal{T}r(n_{i-1}) + 1 \; if \; x_{i-1} \neq x_i \\ &\mathcal{T}r(n_i) = \mathcal{T}r(n_{i-1}) \; if \; x_{i-1} = x_i \end{aligned} \tag{3.3}$$

– the walking time is calculated from:

$$\begin{aligned} &\mathcal{W}r(n_i) = \mathcal{W}r(n_{i-1}) + \varphi_{n_{i-1}n_i}(x_{i-1},t_{i-1}) \; if \; x_{i-1} \neq x_i \\ &\mathcal{W}r(n_i) = \mathcal{W}r(n_{i-1}) \; if \; x_{i-1} = x_i \end{aligned} \tag{3.4}$$

– the waiting time at a node n_i is computed from the scheduled departure time t_i and the arrival time r_{n_i}. This is also considered while computing the travel time along the journey. The waiting time at the destination node is equal to 0. At the source node S, the waiting time represents the difference between the departure time of the chosen departing vehicle t_1 and the departure time of the journey dt.

$$\begin{aligned} \Psi^{x_{i-1}x_i}_{n_{i-1}n_i n_{i+1}}(t_i) &= t_i - r_{n_i} \\ \Psi(S) &= t_1 - dt \end{aligned} \tag{3.5}$$

The definition of a route according to Equation 3.1 allows for one node n_i to deduce its successor node n_{i+1}, its predecessor node n_{i-1} and all information associated with them. Therefore, the notations related to the parameters of a route may be respectively represented by $\varphi(n_i), \Psi(n_i), C(n_i)$, $\mathcal{T}r(n_i)$, and $\mathcal{W}r(n_i)$ with:

– $\varphi(n_i)$ the travel time of the journey starting from n_i;

– $\Psi(n_i)$ the waiting time at n_i;

– $C(n_i)$ the monetary cost of the journey starting from n_i;

– $\mathcal{T}r(n_i)$ the number of transfers used up to n_i;

– $\mathcal{W}r(n_i)$ the walking time up to n_i;

Each route (or path) is characterized by a length represented by $\Pi_{SD}(t_1)$ where t_1 is the departure time from S. Among all the possible paths between S and D the best path is the one with the smallest length $\Pi^*_{SD}(t_1)$. Determining this length depends on the definition of the function $Z(path)$ that we want to minimize.

$$\Pi_{SD}(t_1) = Z(path) \tag{3.6}$$

The definition of this latter depends on the variation of the routing problem studied. In a single criterion context, the function $Z(path)$ that we want to minimize may be defined according to one of the following representations:

$$Z(path) = f_1(path) = \sum_{n_i \in path - \{D\}} (\Psi(n_i) + \varphi(n_i)) \tag{3.7}$$

where f_1 refers to the waiting time and the travel time of the journey.

$$Z(path) = f_2(path) = \sum_{n_i \in path - \{D\}} C(n_i) \tag{3.8}$$

where f_2 refers to the monetary cost of the journey.

$$Z(path) = f_3(path) = \mathcal{T}r(D) \tag{3.9}$$

where f_3 refers to the number of transfers along the journey.

$$Z(path) = f_4(path) = \mathcal{W}r(D) \tag{3.10}$$

where f_4 refers to the walking time along the journey.

The definition of $Z(path)$ for the biobjective and multiobjective that we will

tackle may be represented as follows:

$$Z(path) = (f_1, f_2)(path) = \{ \sum_{n_i \in path - \{D\}} (\Psi(n_i) + \varphi(n_i)), \sum_{n_i \in path - \{D\}} C(n_i) \} \tag{3.11}$$

$$Z(path) = (f_1, f_3)(path) = \{ \sum_{n_i \in path - \{D\}} (\Psi(n_i) + \varphi(n_i)), \mathcal{T}r(D) \} \tag{3.12}$$

$$Z(path) = (f_2, f_3)(path) = \{ \sum_{n_i \in path - \{D\}} C(n_i), \mathcal{T}r(D) \} \tag{3.13}$$

$$Z(path) = (f_1, f_2, f_3)(path) = \{ \sum_{n_i \in path - \{D\}} (\Psi(n_i) + \varphi(n_i)), \sum_{n_i \in path - \{D\}} C(n_i), \mathcal{T}r(D) \} \tag{3.14}$$

$$Z(path) = (f_1, f_2, f_3, f_4)(path) = \{ \sum_{n_i \in path - \{D\}} (\Psi(n_i) + \varphi(n_i)), \sum_{n_i \in path - \{D\}} C(n_i), \mathcal{T}r(D), \mathcal{W}r(D) \} \tag{3.15}$$

Finally, the constraints related to the problem may be represented as follows:

– the effective departure time at the origin departure node S has to be greater

or equal to the departure time of the routing request:

$$t_1 \geq dt \tag{3.16}$$

– the departure time at a node n_i has to be equal to the sum of the arrival time and the waiting time at n_i:

$$t_i = r_{n_i} + \Psi(n_i) \tag{3.17}$$

– the arrival time at a node n_i has to be equal to the sum of the departure time t_{i-1} at the node n_{i-1} and the travel time $\varphi_{(n_{i-1}n_i)}$ of the edge (n_{i-1},n_i) with respect to t_{i-1}:

$$r_{n_i} = t_{i-1} + \varphi_{(n_{i-1}n_i)}(x_{i-1}, t_{i-1}) \tag{3.18}$$

– the departure time, the arrival time, the waiting time and the travel time have to be positive: $t_i \geq 0$, $r_{n_i} \geq 0$, $\Psi(n_i) \geq 0$, $\varphi(n_i) \geq 0$

3.5 Synthesis

We have proposed in this chapter a modelling approach for representing a multimodal transportation network. Both individual and collective transport modes are taken into consideration. The static, dynamic and stochastic aspects of the transport system are also considered. Besides, the proposed approach allows handling multiple criteria shortest path problems and other advanced routing issues. The multimodality was considered by building a separate digraph for each transport mode, and then using the pedestrian network as an intermediate network to transfer from one mode to another. Car parkings and bike rental stations are used as transfer points for individual transport modes. Handling the dynamic nature of networks is done by using time dependent edges. The cost of each edge depends on the time at which the edge is crossed. To account for uncertainty, time dependent distribution functions have been used. The travel time is no longer a time variant static value; it is however, a random variable that may vary within a range according to a custom distribution function. After graphically describing the proposed model, we introduced its corresponding mathematical formulation. The decisions variables, the set of constraints and the objective functions have been formally described. We present in next chapter how the modelling approach can be exploited to solve basic and advanced routing issues. We also highlight later in this manuscript the high amount of flexibility which characterizes this modelling approach in order to deal with several variants of routing problems.

Chapter 4

Exact Algorithms for Route Planning in Deterministic Multimodal Networks

*We present in this chapter a generic routing algorithm that can compute single and multiple criteria shortest paths in a multimodal network. This latter has a time dependency property in the sense that each edge has a cost that depends on the time the edge is crossed. The routing request to be solved consists of a departure place (*e.g. *station, platform, parking, point of interest, etc.), an arrival place and a departure time. We introduce in section 4.2 the pseudo code of the algorithm, and we devote section 4.3 for experimentations.*

Sommaire

Related publications:

1. Omar Dib and Marie-Ange Manier, Laurent Moalic and Alexandre Caminada. Combining VNS with Genetic Algorithm to solve the one-to-one routing issue in road networks. COR, Elsevier, 2015. (Impact Factor: 2.6, SJR: Q1).
2. Omar Dib, Marie-Ange Manier, and Alexandre Caminada. Memetic algorithm for computing shortest paths in multimodal transportation networks. Transportation Research Procedia. Elsevier, 10: 745–755, 2015.
3. Computing multimodal itineraries Watch Video
4. Rerouting in case of perturbation w.r.t vehicles capacities Watch Video

4.1 Introduction

After presenting the modelling approach in the previous chapter, we introduce in this chapter a generic multicriteria shortest path algorithm to compute the set of Pareto journeys with respect to some criteria between two places.

Although many algorithms have been proposed in the literature for computing Pareto-optimal paths, applying such approaches as they are on the proposed model would not solve the problem. As previously mentioned, the proposed model deals with more realistic issues. Thereby, computing multicriteria shortest paths requires additional adaptations.

The emerging problem consists of determining the entire set of nondominated paths to go from one place (*i.e.* station, parking, address) at certain departure time to another place with respect to some criteria. Although scheduled based transportation modes have a time dependency stems from timetable information, the proposed modelling approach allows solving the problem as a variant of the shortest path problem. Criteria that are considered in this chapter are: the travel time f_1, the monetary cost f_2, and the number of transfers f_3. It is important to mention that switching from one mode to the same one (*e.g.* railway to railway) is also counted as a transfer.

The formula related to the computation of each of these objectives are respectively presented as follows:

$$Z(path) = f_1(path) = \sum_{n_i \in path - \{D\}} (\Psi(n_i) + \varphi(n_i)) \tag{4.1}$$

the objective is to minimize f_1 which represents the waiting time and the travel time of the journey. The reader can also refer to Equation 3.7 on page 52 for more details.

$$Z(path) = f_2(path) = \sum_{n_i \in path - \{D\}} C(n_i) \tag{4.2}$$

the objective is to minimize f_2 which represents the monetary cost of the journey. The reader can also refer to Equation 3.8 on page 52 for more details.

$$Z(path) = f_3(path) = \mathcal{T}r(D) \tag{4.3}$$

the objective is to minimize f_3 which represents the number of transfers along the journey. The reader can also refer to Equation 3.9 on page 52 for more details.

4.2 Exact Muticriteria Routing Algorithm

In contrast to single criterion shortest path algorithms, the search process of multicriteria algorithms cannot be stopped whenever the destination node is met. Visiting the destination node means that a feasible solution s has been found. However, there is no guarantee that other solutions that dominate s may exist. Therefore, additional search effort has to be done in order to ensure the absence of nondominated solutions. This additional operation is crucial for any multicriteria shortest path algorithm.

Besides, comparing two solutions s and s' in single criterion optimization is quite straightforward. s is better than s' if the criterion's value associated with s is better than s' (in minimisation problems). In multicriteria optimization, comparing solutions turns out to be difficult since several criteria are taken into consideration.

As previously mentioned, a multicriteria optimization problem can be transformed into a single criterion problem by assigning weights to every criterion. Such a method can be very efficient to reduce the running time of the optimization algorithm. However, one single solution will be obtained at each execution. Several executions should thus be made in order to obtain several solutions. Moreover, there is no guarantee that all nondominated solutions can be obtained while varying the values of weights assigned to objectives. Therefore, users will never be provided with an adequate representation of the solutions space.

We present in Algorithm 5 the main steps of the proposed multicriteria optimization algorithm. This latter does a fast enumeration of all solutions, so then it provides the exact optimal solutions of the problem. At first, each node u in the multimodal graph is assigned a list that may contain a set of nondominated solutions from the departure node s to u. A nondominated solution is a data structure containing labels that represent the criteria to be optimized. Several cases exist to assign values to those labels while the routing algorithm accomplishes its search process.

Moreover, each edge in the graph is assigned a set of labels representing the value of criteria to be optimized. While some labels may have static values such as distance, cost, number of transfers, etc., other labels may be assigned values on the fly. For instance, when dealing with the time label, one cannot precompute the time between two platforms belonging to two different stations unless the algorithm reaches the departure platform.

It is worth to mention here that the proposed modelling approach allows handling unlimited number of criteria. The main key point to handle one criterion is to well define criteria's values through each edge on the graph. After emphasizing on the ability of the proposed model in handling the emerging multicriteria routing

problem, we present now the structure of the algorithm itself.

As most shortest path algorithms, the reference is usually the algorithm of Dijkstra. Adaptations are then done according to the routing problem itself. In our case, we use a modified version of Dijkstra's algorithm based on a multilabel setting rather than one single label. As input, the algorithm takes a routing request consisting of departure and arrival places, a departure time and a set of criteria.

The output of the algorithm is one or a set of nondominated journeys that go from the departure place to the arrival one according to the departure time and the criteria to be optimized. The algorithm has three main phases: initialization, relaxation and visualization.

In the initialization phase, a priority queue pq that may contain different types of nodes is declared and initialized with the departure node. In addition, each node u in the multimodal graph is initialized with a list containing one nondominated solution nds. Other nondominated solutions will be added later to the lists by the algorithm.

Labels inside nds that correspond to the criteria are initialized with a big integer value. Labels inside S are, however, initialized differently depending on the criterion itself. If we consider some parameters such as distance, cost, transfers, waiting time, etc., we initialize them with 0. However, for the time label, its value is set to the departure time.

The second phase consists of the relaxation. The algorithm here repeats some operations until it reaches its stopping criterion. In this work, the algorithm terminates when the priority queue pq becomes empty. In contrast to single shortest path algorithm, one node may enter the queue several times and the algorithm cannot stop when it reaches the arrival node since at this stage, there is no guarantee that no other solutions exist.

In the main loop, the algorithm takes the first element n in the queue pq. Its adjacent nodes are then investigated one by one with respect to the values of labels stored on each adjacent edge. The investigation is done as follows: for every nondominated solution in the list of n, a candidate nondominated solution ca is constructed with respect to labels of each adjacent edge.

A comparison is then made between ca and each nondominated solution $nds2$ of the adjacent node. The algorithm adds ca to the list of nondominated solutions if it is not dominated by any solution in the list. Otherwise, it is not added. In addition, we remove each solution from the list that are dominated by ca. After that, the algorithm removes and adds the adjacent node to the queue if modifications have been made to its nondominated list. Insertion and removing elements from the queue are done in a predefined lexicographical order.

Algorithm 5: An exact algorithm for solving the MOSP

input : Departure place S, Arrival place D, Departure time dt
A set of criteria $C = \{c_1, c_2, \ldots, c_n\}$
output: Set of nondominated paths from S to D w.r.t. dt and C

```
1  begin
       // Iniatilization
2      Initialize a priority queue pq
3      Initialize the list of nondominated solutions of S
4      Initialize the list of nondominated solutions of all other nodes
5      Add S to pq
       // Relaxation
6      while pq is not empty do
7          Node n = pq.poll()
8          l1{} = n.getNonDominatedSolutions()
9          foreach nds1 ∈ l1 do
10             la{} = n.getAdjacentElements()
11             foreach a ∈ la do
12                 ca = build(n,a.getEdge())
13                 added = false
14                 l2{} = a.getNode().getNonDominatedSolutions()
15                 foreach nds2 ∈ l2 do
16                     if ca.dominates(nds2) then
17                         l2.remove(nds2)
18                         added = true
19                     else if nds2.dominates(ca) then
20                         added = false
21                     else
22                         added = true // nds2&ca are nondominated
                           solutions
23
24                     end
25                 end
26             end
27             if added = true then
28                 l2.add(ca)
29                 pq.remove(a.getNode())
30                 pq.add(a.getNode())
31             end
32         end
33     end
34 end
       // Visualisation
35 foreach Itinerary i ∈ T.getNonDominatedSolutions() do
36     Path p = i.reconstructPath()
37     Visualize(p);
38 end
39 end
```

The last step is to retrieve paths found. As at each step the algorithm maintains a history vector about the search process, a backward search is accomplished for retrieving paths. The procedure starts with the target node and goes backward until the departure point is reached.

Finally, it is worth to mention that the proposed generic algorithm can be adapted to deal with other basic and advanced routing issues. For instance, constraints could be easily added on the type of next adjacent nodes to provide routes that exclude some transport modes.

In addition, small adaptations can be applied to solve the latest departure problem that consists of providing the latest departure time to go from one place and arrive at another place at certain time. Such problem can be solved by making the algorithm starts by the arrival stations and adding some constraints related to the arrival time.

Example: In this example, we apply the multicriteria routing algorithm explained in the previous section to find the set of Pareto optimal solutions to go from one station to another in a simplified collective transit network. We show in Figure 4.1 the network for which the algorithm is applied. The network consists of three stations; five platforms; nine events and two transfer edges.

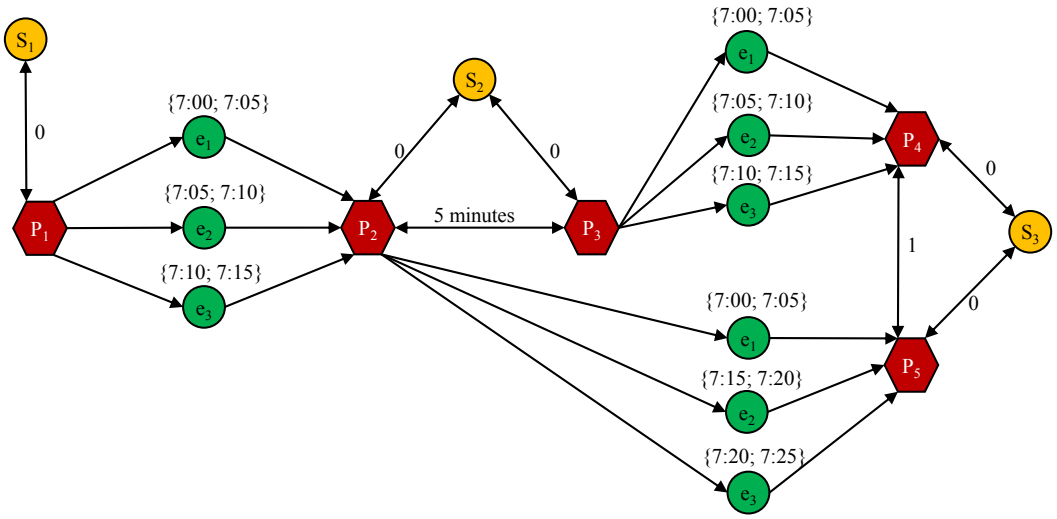


Figure 4.1 – An example of a collective transit network

It is assumed in this example that the time required to go from one station to one of its platforms is set to zero. The request we want to handle consists of finding nondominated paths to go from the station S_1 at the departure time 7:00 to arrive to the station S_3. Numbers in brackets represent departure and arrival times of vehicles departing from one platform to another. The criteria we want to optimize

are: the travel time, the number of transfers and the total walking time.

In Table 4.1, we show the iterations of the proposed algorithm. We present at each iteration the state of platforms, as well as, the elements existing in the priority queue. We also highlight in the blue color platforms that enter the priority queue after each iteration. The following attributes stored in each platform are updated at each iteration:

- Nds: the list of nondominated solutions that have been found between the departure station and the corresponding platform.
- at: the arrival time at the corresponding platform.
- nt: the number of transfers; this attribute corresponds to the value of the criterion related to the number of transfers.
- wkt: the value of the total walking time criterion.
- $fromPlat$: a reference to the platform from which the nondominated solution is created. This is used to retrieve the path in order to visualize it.
- $fromSol$: This attribute is also used when retrieving the path and it accounts for the solution from which the current nondominaed solution has been created.

As can be noticed from Table 4.1, after six iterations, the algorithm terminates and results in two nondominated paths according to the user's request.

As noticed from the table, the solutions found are incomparable. While the itinerary I_1 is better than the I_2 regarding the travel time criterion, the itinerary I_2 dominates I_1 regarding the number of transfers and the total walking time.

To retrieve itineraries found, the algorithm visits each platform belonging to the arrival station and accomplishes a backward search based on the values of both references $fromPlat$ and $fromSol$ until the departure station is reached. By doing so, the total final path will be known and the value of different criteria at each platform will also be shown.

Table 4.1 – Example of multicriteria routing

Iteration	Platforms Objects
1	$P_1 : Nds = [\{at = 7:00, nt = 0, wkt = 0, fromPlat = null, fromSol = -1\}]$ $P_2 : Nds = [\{at = \infty, nt = \infty, wkt = \infty, fromPlat = null, fromSol = -1\}]$ $P_3 : Nds = [\{at = \infty, nt = \infty, wkt = \infty, fromPlat = null, fromSol = -1\}]$ $P_4 : Nds = [\{at = \infty, nt = \infty, wkt = \infty, fromPlat = null, fromSol = -1\}]$ $P_5 : Nds = [\{at = \infty, nt = \infty, wkt = \infty, fromPlat = null, fromSol = -1\}]$ **PriorityQueue queue** = $\{P_1\}$
2	$P_1 : Nds = [\{at = 7:00, nt = 0, wkt = 0, fromPlat = null, fromSol = -1\}]$ $P_2 : Nds = [\{at = 7:05, nt = 0, wkt = 0, fromPlat = P_1, fromSol = 0\}]$ $P_3 : Nds = [\{at = \infty, nt = \infty, wkt = \infty, fromPlat = null, fromSol = -1\}]$ $P_4 : Nds = [\{at = \infty, nt = \infty, wkt = \infty, fromPlat = null, fromSol = -1\}]$ $P_5 : Nds = [\{at = \infty, nt = \infty, wkt = \infty, fromPlat = null, fromSol = -1\}]$ **PriorityQueue queue** = $\{P_2\}$
3	$P_1 : Nds = [\{at = 7:00, nt = 0, wkt = 0, fromPlat = null, fromSol = -1\}]$ $P_2 : Nds = [\{at = 7:05, nt = 0, wkt = 0, fromPlat = P_1, fromSol = 0\}]$ $P_3 : Nds = [\{at = 7:10, nt = 1, wkt = 5minutes, fromPlat = P_2, fromSol = 0\}]$ $P_4 : Nds = [\{at = \infty, nt = \infty, wkt = \infty, fromPlat = null, fromSol = -1\}]$ $P_5 : Nds = [\{at = 7:20, nt = 0, wkt = 0, fromPlat = P_2, fromSol = 0\}]$ **PriorityQueue queue** = $\{P_3, P_5\}$
4	$P_1 : Nds = [\{at = 7:00, nt = 0, wkt = 0, fromPlat = null, fromSol = -1\}]$ $P_2 : Nds = [\{at = 7:05, nt = 0, wkt = 0, fromPlat = P_1, fromSol = 0\}]$ $P_3 : Nds = [\{at = 7:10, nt = 1, wkt = 5minutes, fromPlat = P_2, fromSol = 0\}]$ $P_4 : Nds = [\{at = 7:15, nt = 1, wkt = 5minutes, fromPlat = P_3, fromSol = 0\}]$ $P_5 : Nds = [\{at = 7:20, nt = 0, wkt = 0, fromPlat = P_2, fromSol = 0\}]$ **PriorityQueue queue** = $\{P4, P5\}$
5	$P_1 : Nds = [\{at = 7:00, nt = 0, wkt = 0, fromPlat = null, fromSol = -1\}]$ $P_2 : Nds = [\{at = 7:05, nt = 0, wkt = 0, fromPlat = P_1, fromSol = 0\}]$ $P_3 : Nds = [\{at = 7:10, nt = 1, wkt = 5minutes, fromPlat = P_2, fromSol = 0\}]$ $P_4 : Nds = [\{at = 7:15, nt = 1, wkt = 5minutes, fromPlat = P_3, fromSol = 0\}]$ $P_5 : Nds = [\{at = 7:20, nt = 0, wkt = 0, fromPlat = P_2, fromSol = 0\}]$ **PriorityQueue queue** =$\{P_5\}$
6	$P_1 : Nds = [\{at = 7:00, nt = 0, wkt = 0, fromPlat = null, fromSol = -1\}]$ $P_2 : Nds = [\{at = 7:05, nt = 0, wkt = 0, fromPlat = P_1, fromSol = 0\}]$ $P_3 : Nds = [\{at = 7:10, nt = 1, wkt = 5minutes, fromPlat = P_2, fromSol = 0\}]$ $P_4 : Nds = [\{at = 7:15, nt = 1, wkt = 5minutes, fromPlat = P_3, fromSol = 0\}]$ $P_5 : Nds = [\{at = 7:20, nt = 0, wkt = 0, fromPlat = P_2, fromSol = 0\}]$ **PriorityQueue queue** = $\{\}$
Output:	Itinerary $I_1 : S_1 \rightarrow P_1 \rightarrow e_1 \rightarrow P_2 \rightarrow P_3 \rightarrow e_3 \rightarrow P_4 \rightarrow S_3$ (at=7:15; nt =1; wkt=5 minutes) Itinerary $I_2 : S_1 \rightarrow P_1 \rightarrow e_1 \rightarrow P_2 \rightarrow e_2 \rightarrow P_5 \rightarrow S_3$ (at=7:20; nt =0; wkt=0 minutes)

4.3 Experimental Results

To assess the performance of the proposed model, an advanced web-based routing application is developed on the real data of the French Region Ile-de-France that includes the city of Paris and its suburbs.

A Experimental Setup

Data for collective modes are provided by the transport organization authority that controls the Paris collective transport network and coordinates the different transport companies operating in Île-de-France, mainly the RATP, the SNCF and Optile. These data comprise geographical information, as well as timetable information for four collective transport modes, which are bus, metro, railway, and tram.

More precisely, data encompass 17,950 stations; 41,047 platforms; 195,000 transfers; 303,000 trips and 6,800,000 events for one day. Original data are provided as text file with respect to the General Transit Feed Specification (GTFS) format.

Another set of data relates to individual transport modes. We focus our experimentations on pedestrian, bike and car sharing. The pedestrian network includes 275,606 nodes and 751,144 edges. The bike network consists of 250,206 nodes and 583,186 edges. The road network encompasses 613,972 nodes and 1,273,170 edges. Data for private modes are taken from Open Street Map. Travel times on the road network are provided by a local transport company for one day. For bike and pedestrian networks, we assume that the walking speed is 4 km/h for pedestrians and 12 km/h for cyclists.

For more information about the geographical perimeter of this study, we show in Figure 4.2 a screenshot taken from Google Maps API for the French Region of Ile-de-France.

Experimentations have been done using Java/Eclipse. The associated runs for solving test problems were performed on an Intel core I5 of 8 GB of RAM. Efficient and appropriate data structures have been used to guarantee fast access to information when needed and thereby improve the performance of routing algorithms.

Before applying routing algorithms, it is important to clarify that the generation time of the different subnetworks does not constitute a bottleneck for the performance of the proposed model. Generating networks corresponding to collective modes only takes 15 seconds.

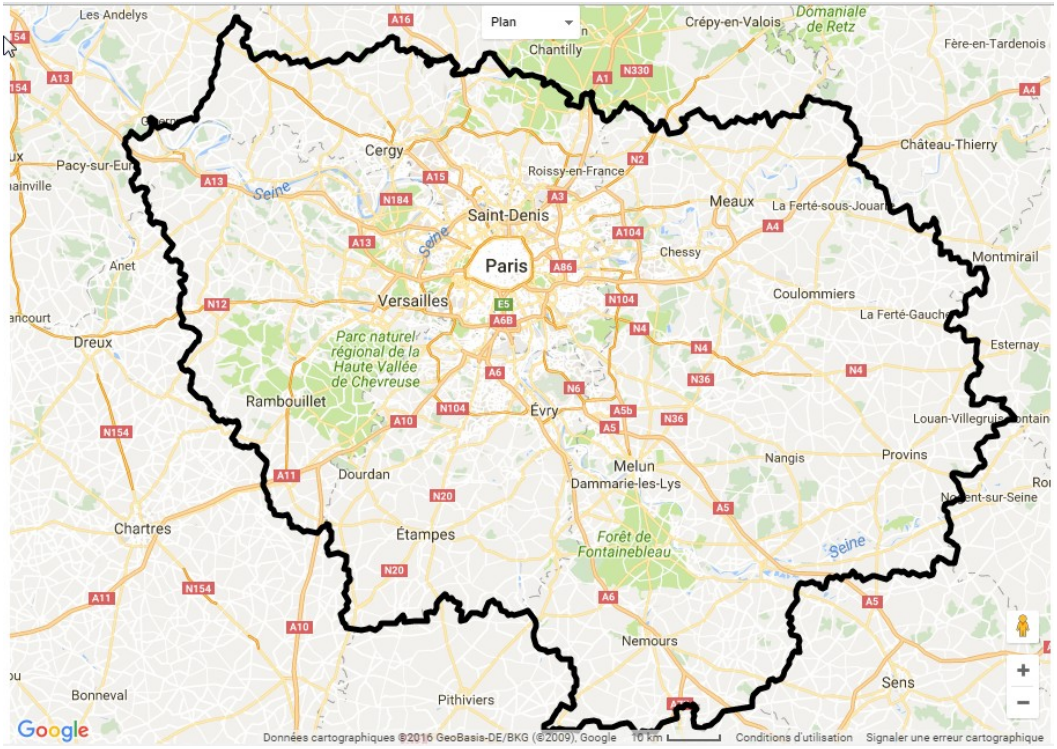


Figure 4.2 – Case study: Ile-de-France Region

Moreover generating individual modes takes less than 5 seconds. However, linking all networks takes a lot of time (> 30 minutes) since for every access point, bike rental and car parking's nodes, the nearest neighbour problem has to be solved with respect to the pedestrian network. To deal with that, a preprocessing operation has been performed whereby nearest neighbours are computed and prepared to be loaded later on when generating networks. By doing so, we ensure a fast generation step for the final multimodal graph.

Since the increased time in computing optimal itineraries decreases the utility of the journey planning services, the computational effort of any routing algorithm constitutes a critical success factor for their integration within an online journey planning decision support system.

To assess the performance of the proposed approach, we have used it to solve 10,000 routing queries. Each request consists of a source and a target node, and a departure time. Those parameters are uniformly generated at random. We assume here that requests allow all transport modes.

Although the proposed routing algorithm is not limited to specific criteria, experimentations are done with respect to the following criteria: the travel time, the monetary cost and the number of transfers. One can add more criteria such as

distance or CO_2 emission by charging their values on edges.

It is worth clarifying that we do not focus in our experimentations on solving routing queries where a private car or a bike is available at the departure point. Bikes and cars can only be taken from rental stations.

In order to allow computing journeys with respect to the defined criteria, labels related to each criterion should be well defined over each edge at the generation phase of the graph. Although assigning costs is very dependent on transport operators, it is assumed in this work that costs are computed according to two parameters: the transport mode and the travelled distance. Therefore, at the generation phase of the model, the cost of each edge is computed according to its length in meters and the transport mode it belongs to.

B Experimental Analysis

We start evaluating the proposed approach by limiting the routing query to only one criterion (*i.e.* we start with the function f_1 (Equation 3.7) on travel time). That is, we want to ensure that the proposed model allows computing multimodal routes in a single criterion context and the proposed algorithm provides correct results. We remind that the result is proved to be optimal and only one run is sufficient for every test configuration.

Analysing and comparing results with real-world routing applications have verified the ability of the proposed model in providing correct itineraries when applying the routing algorithm.

When it comes to assessing the computational performance, results indicate that the algorithm does not exceed the ratio of 300 milliseconds to handle a request (Table 4.2). That also remains true even when we change the criterion to be the cost (*i.e.* the function f_2 (Equation 3.8)) or the number of transfers (*i.e.* the function f_3 (Equation 3.9)).

We have also noticed that the computational time increases with the number of networks used. By deactivating one mode (*e.g.* bus, road, bike), the running time for answering a routing query decreases. This can be explained by the fact that the search space of the routing algorithm increases with the size of the final generated graph. More networks means more nodes and edges, thus more works for the search algorithm.

Results in Table 4.2 also indicate that when changing the criterion, the algorithm's search space does not relatively change.

Table 4.2 – Running time for the single criterion experimentations on 10,000 routing queries

	Criterion to minimize					
	f_1 **time**		f_2 **cost**		f_3 **transfers**	
Modes	◊	□	◊	□	◊	□
a	90	115	98	118	94	118
b	140	170	147	171	135	175
ab	257	295	255	292	252	298

◊ average execution time in milliseconds (ms) □ maximal execution time in ms
f_1: travel time f_2: monetary cost f_3: number of transfers
a: individual modes b: collective modes ab: individual & collective modes

After proving the efficiency of the proposed approach in a single criterion context, its behaviour in a multicriteria context is now analysed. While computing routes in a single criterion context was done without any limitations, computing Pareto routes tends to be more challenging.

Results indicate that Pareto front's size may grow in some cases very rapidly even when only two criteria are considered. However, in other cases, the algorithm only results in few routes. While the average final set of nondominated solutions when considering two criteria is five, it increases to 14 when considering three criteria. Again the final set is guaranteed to be Pareto optimal.

When it comes to the computational performance, results in Table 4.3 show that the maximal running time in two criteria context does not exceed 6 seconds. It, however, increases to 32 seconds when considering 3 criteria. Thus, it is obvious that the proposed approach cannot be used in its actual version to support online users' queries.

As a result, enhancements have to be made in order to accelerate the search process of the routing algorithm. This can be usually done by preprocessing some data in the offline mode in order to exploit them when receiving online routing queries. Such techniques are very efficient in static networks, however, applying them in the transport field remains a difficult task due to the dynamic nature of transport network. Results also show that the average running time and the average number of nondominated solutions proportionally vary with the number of criteria. More criteria leads to an increase in the search space and thereby, an increase in the running time. By analysing the time required to accomplish each iteration in the algorithm, we observed that adding elements to the queue is the main bottleneck for the computational time (lines 26 to 30 of the algorithm). Therefore, it is crucial to apply strategies whereby we prevent unimportant nodes from entering the queue. That is, we have to apply strategies to reduce the algorithm's search space and thereby enhance its running time.

Table 4.3 – Running time for the multicriteria experimentations on 10,000 routing queries

Criteria to minimize								
	(f_1,f_2)		(f_1,f_3)		(f_2,f_3)		(f_2,f_2,f_3)	
Modes	◊	□	◊	□	◊	□	◊	□
a	1.1	1.9	0.9	2	1.3	2.6	8.6	10
b	1.4	2.9	1.3	2.9	1.9	2.7	15	20.9
ab	3	5	2.9	4.8	3	5.2	25	32

◊ average execution time in seconds □ maximal execution time in seconds
f_1: travel time f_2: monetary cost f_3: number of transfers
a: individual modes *b*: collective modes *ab*: individual & collective modes

Analysing some nondominated solutions has also shown that the algorithm may result in unimportant journeys for users. For instance, no one is interested in a path including one hour of walking. Therefore, the first strategy we apply (named s_1) is to put upper limit bounds for the number of transfers. In a similar word, we say here that as a passenger, I am only interested in journeys that do not exceed this limit of transfer. We reaccomplished experimentations by putting 6 as an upper limit bound for the number of transfers.

Results presented in Table 4.4 have indicated that the performance of the algorithm has relatively enhanced after applying the strategy s_1, but it still not applicable for users seeking fast answers. Upper bound limits in this strategy can vary according to the behaviours of users; using the above numbers is a constraint made by the transport operator we are working with. Therefore, adaptations are required when using this strategy to deal with routing problems in other regions. Such adaptations should be based on statistics and observations as well as, advanced models describing the passengers' behaviours towards such criteria.

A second strategy (named s_2) that may be very efficient is to enhance the nondomination relationship. After analysing the results, we saw that although two solutions are nondominated in theory, only one solution should be kept in practice. For example, let us assume a path p_1 with 15 minutes of travel time and 5 transfers. p_2 is another solution with 40 minutes travel time and 4 transfers. Theoretically speaking, p_1 and p_2 are nondominated solutions. However, in practice, one would not choose an itinerary with more than 20 minutes in comparison with another path to only gain one transfer. That is, it is not important to keep one solution that slightly enhances one criterion while extremely worsen another criterion. That is a process oftenly used in multicriteria optimization and defined as fitness sharing introduced by Goldberg and Richardson [**?**] with the computation of phenotype distance in the selection space of the function to optimize. To resume it, the fitness of each individual is scaled based on its proximity to others, and kept or not.

To reaccomplish experimentations we have set the difference between criteria's values for two compared solutions as follows: 20 minutes for travel time; 5 euros for the cost; 2 for the number of transfers. These parameters can be optimally set after investigating the behaviours of passengers. However, in this work, we set them according to the preferences of the transport operator we are working with. Such numbers are thus based on real world statistics and observations and are parts of the use case we are trying to address.

After applying both s_1 and s_2, results from Table 4.4 indicate that the performance of the proposed approach has been vastly improved in comparison with the standard approach where no fitness sharing is applied. Although we have succeeded in reducing the average running time, the proposed approach requires further improvements to be integrated within an online routing system. Therefore, we introduce a third enhancement strategy (named s_3). Indeed, as we have said previously, in a single criterion context, one can abort the search once the algorithm reaches the arrival node. This is not true in a multiobjective context. Therefore, the arrival node can be visited several times before the algorithm terminates. Here we have remarked that once the arrival node is visited at least once, we can benefit from it in the search process. In other words, once we find a nondominated solution *nds* between the departure and arrival node, we start using the *nds* criteria's values as upper bounds for the continuation of the search. By doing so, we avoid unimportant nodes much earlier in the search phase. Bounds will be enhanced much more when additional nondominated solutions are found. In Table 4.4, we present experimental results and level of enhancements after applying different accelerating strategies (s_1, s_2, s_3) with respect to the standard approach (s) where no enhancement is applied. Both individual and collective modes are considered in queries.

Table 4.4 – Impact of enhancement strategies on running time for individual and collective modes together

	Criteria to minimize							
	(f_1,f_2)		(f_1,f_3)		(f_2,f_3)		(f_1,f_2,f_3)	
Strategies	◊	□	◊	□	◊	□	◊	□
s	3	5	2.9	4.8	4.7	5.2	25	32
s_1	2.4	2.6	2.6	2.9	2.3	2.7	19	28
s_1 & s_2	1.1	1.3	1.2	1.8	1	1.4	15	23
s_1 & s_2 & s_3	0.9	0.7	0.8	0.7	0.8	1	12	19

◊ average execution time in seconds □ maximal execution time in seconds
f_1: travel time f_2: monetary cost f_3: number of transfers
s: basic Algorithm 5 (A5) s_1: A5 with fixed upper bound on transfer s_2: A5 with fitness sharing
s_3: A5 with dynamic upper bound on all criteria

As can be observed from Table 4.4 above, after applying all strategies, we have succeeded in reducing the average running time in the worst case (*i.e.* when 3 cri-

teria are considered) from 25 to 12 seconds. Moreover, the worst case has improved to reach 19 seconds. In contrast to the running time, the average number of non-dominated solutions has not largely changed. While the average Pareto front size decreases from 5 to 3 in a two criteria environment, it decreases from 14 to 9 when considering three criteria.

C Real Case Scenario

To have a better evaluation of the proposed work, an advanced web routing application has been developed. The user interface has been developed using HTML, JavaScript, and JQuery. Our algorithm has been implemented in Java. To visualize itineraries we have used the Google Maps API.

We show in Figure 4.4, a real case scenario: it consists of a request to go from the station "Tour Eiffel" to the station "La defense", which are both located in the Region Ile-de-France. The departure time is "10:45". Criteria considered are the travel time, the monetary cost, the number of transfers and the walking time; see all the functions f_1 to f_4 defined in Chapter 3. The modes used are bus, railway, tram and metro. After applying the proposed algorithm, three nondominated paths have been found. In Figure 4.3, we present the stations constructing such paths, as well as, the values of the non-dominated paths. We present in Figure 4.4, the user interface of the developed routing application and the results of the previously mentioned query. Three paths among many others are presented and visualized using the Google Maps API.

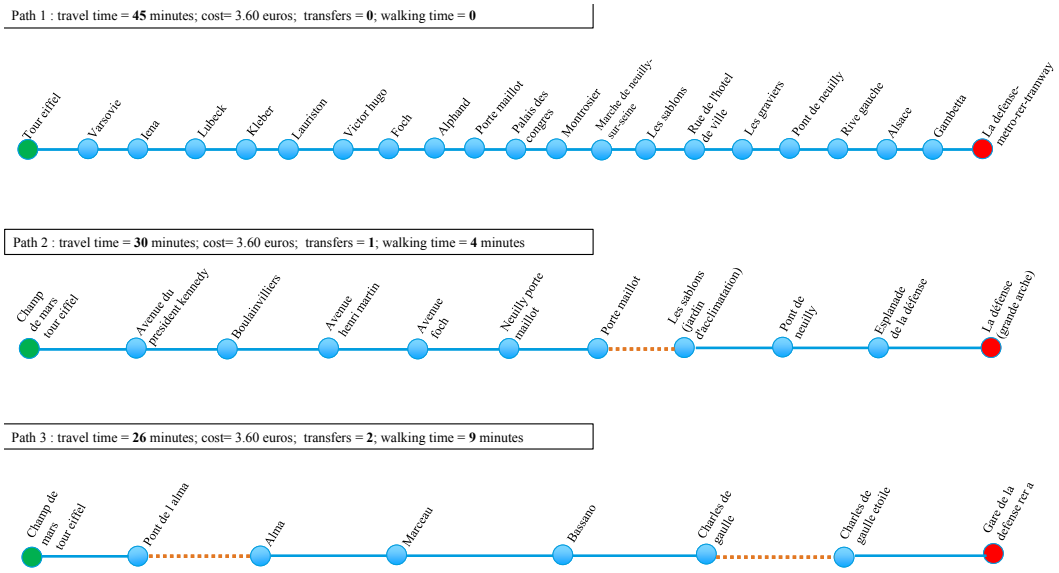


Figure 4.3 – Example of three nondominated paths found while going from the station "tour Eiffel" to "La defense"

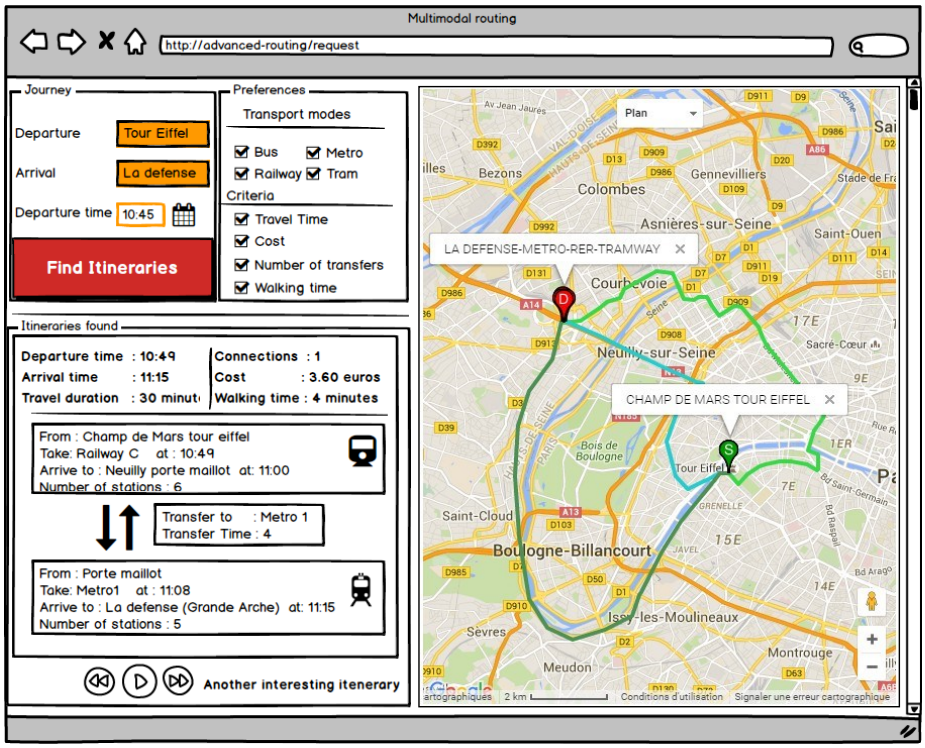


Figure 4.4 – Advanced web-based routing application

4.4 Synthesis

A multicriteria exact routing algorithm was proposed in this chapter. The algorithm can compute the full set of nondominated solutions between two places with respect to a departure time and a set of criteria. The algorithm is not limited to our set of four criteria and also an unlimited number of transportation modes can be taken into consideration. Time dependent edges were considered and restrictions on the capacity of vehicles were handled. Experimentation have been made by developing a web-based routing algorithm for the French Region Ile-de-France. The results indicated that the running of the proposed algorithm hinders its usage in real world route planning system. To overcome it, several attempts have been made in order to develop accelerating techniques so that the algorithm's search space decreases. Although the results indicate that good improvements can be achieved using the proposed accelerating strategies, however, there is no guarantee about their performance on very large multimodal networks such that in Ile-de-France Region. Therefore, we propose in next chapter some advanced metaheuristics whereby routing issues are solved within acceptable amount of computational time.

Chapter 5

Route Planning with Metaheuristics

As previously indicated, using an exact algorithm for computing the set of nondominated solution to go from one place to another with respect to a departure time and a set of criteria may not be very efficient due to the running time limitation. Therefore, we aim in this chapter at solving routing issues using metaheuristics. These latter are powerful methods to provide high quality solutions within reasonable amount of computational time. We propose a combination between Genetic Algorithms (GA) that belong to the population based metaheuristics and Variable Neighbourhood Search (VNS) that belongs to the single solution metaheuristics.

Sommaire

Related publications:

1. Omar Dib, Laurent Moalic, Marie-Ange Manier, Alexandre Caminada. An advanced GA–VNS combination for multicriteria route planning in public transit networks. Expert Systems with Applications, Pergamon, 72: 67–82, 2017. (Impact Factor: 3.928, SJR: Q1, CORE: B).

2. Omar Dib, Alexandre Caminada, and Marie-Ange Manier. A Genetic Algorithm for solving the multicriteria routing problem in public transit networks. 6th international conference on metaheuristics and nature inspired computing META , October 27-31 marrakech, META16, 2016.

5.1 Introduction

Solving optimization problems using metaheuristics usually requires passing through several steps. Firstly, decisions variables of the optimization problem have to be defined. For instance, in the proposed, what has to be decided is whether a node is included in the final solution or not. Here the decision variable may take a binary value 0 or 1.

Secondly, constraints related to the problem have to be identified. For instance, in the studied routing problem, edges in the final solution have to be related. The tail of each edge has to be the same as the head of the predecessor edge except for the departure and arrival stations. Thirdly, an objective function has to be built. In the studied problem, the aim is to minimize one single criterion such as the travel time and number of transfers, or a set of criteria at the same time.

After that, representing a solution for the studied problem has to be proposed. Several representations exist in the literature in order to encode a solution such as binary or permutation encodings. Choosing an encoding scheme is crucial for the resolution phase since it directly affects the search process of the algorithm. In addition, some representations cannot always be adapted for all optimization problems.

Another important step is to define a neighbouring structure. In this latter, we answer the question "How to go from one solution to another? Choosing a neighbouring structure has a direct effect on the performance of the optimization algorithm. After that, specific steps and parameters have to be defined for each metaheuristic. For instance, genetic operators such as selection, crossover and mutation have to be defined for Genetic Algorithms and several neighbouring structures have to be defined for the Variable Neighbourhood Search.

The studied routing problem in this chapter consists in using metaheuristics to solve the multicriteria shortest path problem in a time dependent multimodal network. Criteria that are considered in this chapter are: the travel time f_1, the monetary cost f_2, the number of transfers f_3, and the walking time f_4.

The formula related to the computation of each of these objectives are respectively presented as follows:

$$Z(path) = f_1(path) = \sum_{n_i \in path - \{D\}} (\Psi(n_i) + \varphi(n_i)) \tag{5.1}$$

the objective is to minimize f_1 which represents the waiting time and the travel time of the journey. The reader can also refer to Equation 3.7 on page 52 for more

details.

$$Z(path) = f_2(path) = \sum_{n_i \in path - \{D\}} C(n_i) \tag{5.2}$$

the objective is to minimize f_2 which represents the monetary cost of the journey. The reader can also refer to Equation 3.8 on page 52 for more details.

$$Z(path) = f_3(path) = \mathcal{T}r(D) \tag{5.3}$$

the objective is to minimize f_3 which represents the number of transfers along the journey. The reader can also refer to Equation 3.9 on page 52 for more details.

$$Z(path) = f_4(path) = \mathcal{W}r(D) \tag{5.4}$$

the objective is to minimize f_4 which represents the walking time along the journey. The reader can also refer to Equation 3.10 on page 52 for more details.

In next section, we present in details the various steps of the proposed Memetic Algorithm, as well as, steps of the Genetic Algorithm and the Variable Neighbourhood Search that are proposed in order to solve the emerging routing problem.

5.2 Memetic Algorithm for Multicriteria Routing

As aforementioned, the main contribution of this chapter is to adapt and apply a heuristic method, which is based on a collaboration between two metaheuristics, for solving the Multiobjective Shortest Path problem. The metaheuristics used are GA that belongs to the population-based algorithms and VNS that belongs to the single point search algorithm.

The proposed method proceeds with a population of solutions as standard GAs works. However, in the proposed approach, initial solutions are generated using a double search algorithm that is able to provide a set of feasible paths between any two nodes. The details of this algorithm is described later in this chapter. Once a population of solutions (feasible paths) is successfully generated, an enhancement operation is accomplished over the first population. That is, a VNS is applied over each individual belonging to the first population. It is decided to improve the initial solutions since their quality would possibly help the algorithm to approach from the

optimal region within the search space. The VNS is applied in such a way that its resulting solution will be feasible. More details about the VNS adaptation to the problem will be discussed later. Once the VNS is performed, individuals are sent to an evaluation process.

To evaluate an individual, a fitness function is computed using the Weighted Average Ranking (WAR) technique [**?**]. This technique makes a good evaluation of the rank of each individual with respect to each criterion of the MOSP. After the improving phase, several genetic operations are performed (selection, crossover and mutation) in the goal of increasing the algorithm's chance to find better solutions. To begin with, a roulette wheel selection has been used as a selection operator. A selection probability is computed according to the WAR associated with each individual. After that, the crossover operation is performed. In this operation, two individuals are chosen according to the selection operator in order to form new individuals (offsprings). By doing so, a new population is produced having twice the size of the current population. The best half individuals are then selected for the next generation and the rest are ignored. Duplicated individuals are replaced with newly generated chromosomes to avoid reprocessing the same individuals. Single point crossover technique has been used in order to produce offsprings. An intersection node of the multimodal transport network between two individuals is selected among their common nodes to be the crossover point. Current individuals exchange then part of genes with each other before or after the crossover point to generate offsprings. The offsprings remain feasible solutions.

Algorithm 6: Pseudo code of the proposed Memetic Algorithm

Initialization: Initialize feasible population using a double search algorithm
Improve solutions using VNS

```
// Iteration
while Stopping rule is not satisfied do
    Evaluation: Evaluate solutions using WAR
    Archive: Update archive
    Selection: Select parents using roulette wheel
    if rand(0,1) ≤ crossover rate then
        Perform single point crossover                    // Crossover
    end
    if rand(0,1) ≤ mutation rate then
        Execute VNS                                       // Mutation
    end
    Replacement: Select individuals to construct the new population
    Update population
end
return the set of nondominated solutions in the archive
```

After that phase, a new population of solutions is created. Thus, new paths between the origin and the destination point have been probably detected. Therefore, re-applying the VNS procedure to the new population will possibly enhance the paths' qualities. That is, the algorithm performs a special mutation operation based on VNS method over each individual (path) in the population. Thanks to this technique, the algorithm will have more chances to exploit through GA and explore through VNS new regions of the search space. The whole process is repeated until the algorithm reaches the predefined stopping criteria. An archive of nondominated solutions is also used and updated at each generation. We show in Algorithm 6 the pseudo code of the proposed Memetic Algorithm. Each component will be discussed in more details in next paragraphs.

A Encoding

Encoding of individuals is a crucial step in GAs. It largely depends on the property of the problem, and will highly affect the design of genetic operations. Several encoding techniques have been proposed to represent solutions in GAs such as binary and permutation encoding. The binary technique consists of computing and then concatenating the binary coding of each variable in a given individual. For instance, in the studied problem, a route consisting of four stations (2, 4, 9, 3) can be represented by the binary string 0010| 0100| 1001| 0011.

In permutation encoding, variables are concatenated without any transformation. For instance, the previous route (2, 4, 9, 3) is encoded as 2| 4| 9| 3. Since the binary encoding requires additional computation effort due to the transcoding (binary/integer), the permutation encoding is used in this work.

A solution for the studied problem is represented by a list of nodes where each node corresponds to a platform in the collective modes or node in the individual modes. The information stored in each node n are related to the identifier of the platform, its mode and the time at which n is traversed in addition to the value of the different objectives considered.

It is important to mention that the time at which each platform is traversed is computed by adding the traversed time of the predecessor platform to the exact travel time between the two platforms. The travel time between two platforms is computed by subtracting the arrival and departure times of the appropriate departure event. This latter represents the first departing vehicle at the departure platform with respect to the arrival time at the departure platform.

To efficiently identifying the first departure event at a platform w.r.t a departure time, event nodes are ordered w.r.t their departure times during the generation phase of the model. In addition, event nodes are stored in sub-categories according

to predefined periods (*e.g.* the list of departing nodes from 8:00 to 9:00). Thanks to such implementation tricks, identifying the appropriate departure event would be very efficient and would also ensure the feasibility of encoding paths.

Since different solutions may represent different routes, the size of the chromosomes is not static; it varies with the number of nodes constructing the route. Finally, a special node is inserted to the solution encoding to deal with transferring from one mode to another. This special node plays an important role since it stores information related to the transfer (walking time, state of the transfer links...). Thus, the computation of the different criteria especially those related to the number of transfers and walking time would be impacted by such transfer nodes. Furthermore, inserting a special node for transfers would help to efficiently identify crossover points as well as, to visualize the details of itineraries resulting from the algorithm.

We show in Figure 5.1 a solution for a routing request between the departure station S and the arrival station T. The encoded solution consists of using three modes: railway, metro and bus. The numbers are the identifier of the used platforms.

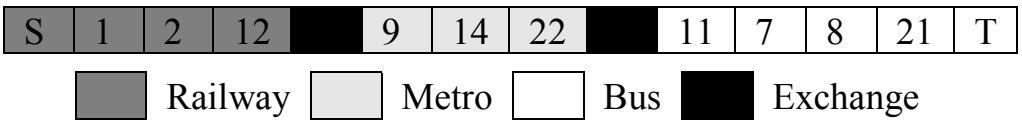


Figure 5.1 – Encoding scheme

B Evaluating Individuals

Several approaches can be used for computing the fitness function of a multiobjective optimization problem. The classical approach is to assign a weight w_i to each normalized objective function $o_i(x)$ so that the problem is converted to a single objective problem with a scalar objective function. While using this approach may largely enhance the computational time to solve the problem, it has the drawback of only providing one single solution. In addition, there is no guarantee that all nondominated solutions can be obtained by varying the weights of criteria.

Therefore, if multiple solutions are desired, the problem should be solved multiple times with different weight combinations. In addition, selecting weight vector for each run remains a difficult issue for users. Furthermore, in such classic approach, there might be a good chance for loosing some interesting solutions after solving the problem. Such solutions may be the best combination of many passengers. Therefore, such standard approaches are not used in this work.

Evaluating individuals in this work has been done using the Weighted Average

Ranking (WAR) technique. Unlike traditional techniques, which are based on a simple scalar value, WAR consists of assigning each individual (route) with a rank that depends on the value of objectives f_i (route). This rank represents the average value of all ranks determined by sorting the current population in a separated manner according to each f_i (route).

It is worth mentioning that by using this evaluation technique, the algorithm is prevented from normalizing all objectives. The normalization phase is not an easy task in the studied problem since ranges of objectives are not known in advance. In addition, the normalization will surely consume additional computational time. Therefore, using WAR instead of traditional evaluation approaches represents an advantage for the proposed algorithm.

More precisely, the evaluation is performed as follows. The population is sorted λ times according to each objective (where λ is the number of objectives considered). At each sorting operation, each individual is assigned a rank value R_i according to its position regarding the objective f_i in the current population. At the end of this operation, each individual will have λ ranks. The global rank GR value will be the average of all ranks associated with an individual.

Table 5.1 – Example of evaluating individuals

Individual	f_1 **score**	f_2 **score**	**Rank according to** f_1	**Rank according to** f_2	**Global rank**
I_1	5	1	1	6	$\frac{7}{2}$
I_2	4	2	2	4	$\frac{6}{2}$
I_3	3.5	1.5	3	5	$\frac{8}{2}$
I_4	2	1.5	5	5	$\frac{10}{2}$
I_5	3	4	4	2	$\frac{6}{2}$
I_6	2	3	5	3	$\frac{8}{2}$
I_7	1	5	6	1	$\frac{7}{2}$

In Table 5.1, the above explained technique is applied to evaluate seven individuals; two criteria are considered in a minimization problem and the smaller the function score the higher the rank of the individual. As can be seen from the table, the global average rank of the individuals I_2 and I_5 is the smallest average $\frac{6}{2}$. This property reflects very well that I_2 and I_5 are dominated by I_4. Since I_4 represents the individual that dominates the maximum number of individuals in the current populations, the WAR strategy results in a high GR for I_4.

C Generating Initial Solutions

Generating initial solutions for metaheuristics is not always straightforward. Usually, the good initial solutions might rapidly guide the search process towards important regions in the search space. The initial solutions in the proposed work are a set of feasible paths generated using a double search algorithm. A double search process that simultaneously run a forward search from the origin point S and a backward search from the destination point T has been used in order to get a set of feasible paths between a pair of nodes.

The algorithm works as follows:

- It initially defines two queues: one for the forward search fq and another for the backward bq. The source and target nodes S and T are then added to fq and bq respectively.
- After initialization, the algorithms starts repeatedly taking the element at the head of fq; adding its adjacent nodes to fq; removing it from fq. The same process is also done from the backward search.
- To keep a history of the search processes, nodes visited from the forward search are assigned an F flag and nodes visited from the backward search are assigned a B flag. Moreover, we store the incoming edges that allow the algorithm to reach each forward node and the adjacent edge of each backward node.
- Once the algorithm is about to add in fq a node that has already been visited from the backward search, it means that a path between S and T has been detected. Therefore, the procedure of constructing that path starts. The same process is also performed from the backward search.
- To reconstruct a path found, the historical information stored in each node object are used.
- The algorithm terminates if one of the queues fq and bq becomes empty. It has been remarked also that in some graph instances the proposed algorithm may provide too many initial paths. Therefore the maximal number of generated paths has been added as another termination criterion for this case. By doing so, the size of the first population can be controlled.

It is worth mentioning that as the algorithm performs, there is a chance that two identical paths are found between S and T. To tackle such problem, the algorithm keeps a history about the intersection edges found during the execution. It then ignores a path if the intersection edge has been already added to the list.

Finally, as can be easily noticed, the process of computing initial solutions mimics the search process used in bidirectional Dijkstra algorithm. However, using the bidirectional Dijkstra in its standard form is not possible since the exact arrival time at the destination node is not known a priori due to the time dependency property of the network.

To overcome this issue, a search process is launched from the destination node with ignoring the exact arrival at the destination node. As a result, a set of feasible paths are obtained with no guarantees about their qualities. Therefore, one can see that the proposed algorithm to compute the initial solutions is a modification of the bidirectional Dijkstra where the exact arrival time at the destination node is ignored and the stopping criterion to get many feasible solutions is adjusted.

D Enhancing Initial Solutions Using VNS

Since the quality of the initial solutions may affect the performance of metaheuristics, we decide to enhance the initial solutions. To do so, a VNS method is used as an enhancement operator. Indeed, a VNS method is applied over the individuals in the first population.

This technique will increase the diversity level of the initial population, as well as, establish a good repartition of solutions within the search space. Consequently, the algorithm's chance to find better solutions will also increase. In the following, the adaptation of the VNS method to deal with the studied problem is explained. Indeed, one of the major challenges addressed in VNS is the construction of the neighborhood structures. To deal with that issue, the proposed algorithm performs a preprocessing operation during the generation phase of the network.

The idea is to examine each edge in the graph and check if its starting and ending nodes share a common node with other edges. Two nodes A and B have a common node if and only if the end point of one adjacent edge of A is the same as the starting point of an incoming edge of B. After doing that, we will end up with two neighboring structures that we will use when applying VNS.

In the graph below, if we take the edge (AC), it can be noticed that the node B is shared between the adjacent edge (AB) of A and the incoming edge (BC) of the node C. Hence, one can deduce that an alternative path can be found to reach the node C from A, which is in this case the path (AB, BC).

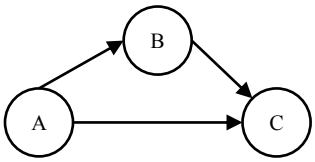


Figure 5.2 – Constructing neighbouring structure

Once such common nodes are found, constructing neighbouring structures can be done as follows. If the path represented by the edge (AC) is dominated by the path (AB, BC), thus, whenever the edge (AC) is met, it should be replaced by the path (AB, BC). That case makes the first neighbouring structure. That is, the first neighbouring structure is a list containing replacement paths formed by two edges for each edge. Else (AC) is kept.

A second scenario may arise if the path (AB, BC) is dominated by the path (AC), thus, the path (AB, BC) can be always replaced by the edge (AC) in any path. The second neighbouring structure is then a list containing an edge that will replace a path formed by two edges.

In the case that (AC) and (AB, BC) are non-dominated, the replacement list of any of them should include the path itself and the other path. By doing so, the solution itself and a nondominated solution belonging to the list of the neighbourhood solutions are maintained.

Figure 5.3 shows a graph with 13 edges and 8 nodes. Only one single criterion is used to facilitate the illustration. An example of the replacements included in the first neighbouring structure is to substitute the edge (SC) with the 2-edges path (SA, AC). Replacing the 2-edges path (BD, DT) by the edge (BT) is an example of an instance existing in the second replacement structure.

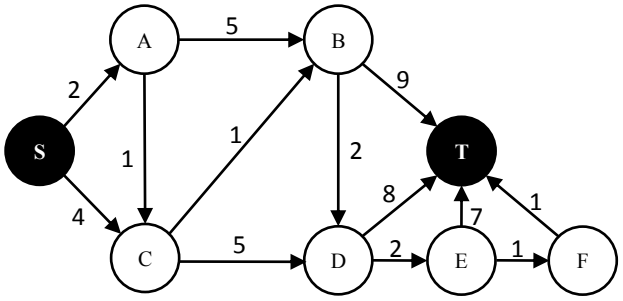


Figure 5.3 – Performing VNS over an individual

After performing the preprocessing operation over the graph above, the algorithm ends up with two replacement lists that illustrate neighbouring structures.

Table 5.2 – Items in the first replacement list

Edge	**Replacements**
(SC)	(SA, AC)
(AB)	(AC, CB)
(CD)	(CB, BD)
(ET)	(EF, FT)
(DT)	(DE, EF, FT)

Table 5.3 – Items from the second replacement list

Path	**Replacements**
(BD, DT)	(BT)
(DE, ET)	(DT)

After carefully examining the two replacement lists, the performance of our VNS can be improved by applying some enhancement operations. More specifically, we try to enhance the quality of solutions that can be found by applying the first structure using information from the two replacements list. For instance, the path (DE, ET) included in the second list can be replaced by the edge (DT). However, if one carefully scrutinizes, it can be noticed that the edge (ET) is included in the first structure and the length of its replacement path added to the length of (DE) is shorter than the length of (DT). Therefore, a new element can be added to the first structure and it will contain the substitution of (DT) by the path (DE, EF, FT). This strategy is only applied in the case that it produces a new dominating solution.

The quality of elements in the first list can also be improved by using the list itself. For instance, let us imagine that the edge (ET) has a weight of 5. The first list will then contain a row containing the replacement of (DT) by (DE, ET). However, (ET) is also in the list. Hence, (DT) can be replaced by (DE, EF, FT). After the enhancement operations, the algorithm gets the chance to find better solutions and thereby its performance will increase. By taking the path $P = (SC, CD, DE, ET)$ which is a solution to go from S to T, the neighbor solutions of P using the first neighbourhood structure are given in Table 5.4:

Table 5.4 – List of solutions with the first neighbouring structure

P_{11}	SA	AC	CD	DE	ET
P_{12}	SC	CB	BD	DE	ET
P_{13}	SC	CD	DE	EF	FT

The neighbour solutions of P after applying the second neighbourhood structure are provided in Table 5.5:

Table 5.5 – List of solutions with the second neighbouring structure

P_{21}	SC	CD	DT

After defining the two neighbouring structures, the mechanism that we adopted to move from one solution to another is explained. As shown in Algorithm 7, the VNS method takes as input a set of neighbouring structures and a single initial solution and it performs then some operations in order to enhance this initial solution.

Although the VNS approach is very popular for its easy adaptation, it turns out that its usage in a multicriteria environment is not an easy task. The difficulty stems from the fact that one solution may not only have one single better solution among all elements in the list of neighbourhood solutions. However, a solution may have several dominated or nondominated solutions in the list of neighbourhood solutions. It is therefore very important for the efficiency of the method to define a strategy whereby the algorithm moves from one solution to a neighbour one during the search process.

To overcome all those difficulties, the VNS is adapted as follows. The VNS proceeds with the first neighbouring structure. It examines then all the neighbours' solutions belonging to the current solution. The algorithm then selects the best neighbour among all neighbours according to the fitness.

To define which neighbour solution is the best one, all neighbourhood solutions are evaluated according to a Weighted Average Ranking (WAR) that we have defined in Section B, the higher the rank the best the solution. VNS then compares the best-selected neighbour with the current solution; it moves to the best neighbour if its rank is better than the rank of the current solution. If not, VNS changes the neighbouring structure. The process is repeated while the current solution can

be enhanced by using any of the neighbouring structures.

Algorithm 7: Pseudo code of the proposed Variable Neighbourhood Search

input : Two neighbouring structures $nk, (k = 1,2)$
An individual (solution) x
output: Best solution found

```
while stopping criterion has not been met do
    k ← 1
    while k ≤ 2 do
        Exploration of neighborhood: find the best neighbor x' of
         x (x' ∈ Nk(x))
        Move or Not:
        if f(x') < f(x) then
            x ← x'
            k ← 1
        else
            k ← k + 1
        end
    end
end
```

Assuming that the path $x = (SC, CB, BD, DT)$ in Algorithm 7 is an initial solution to the VNS method with length 15. To apply VNS, the algorithm performs as follows: It uses the first neighbouring structure and calculates all the neighbour solutions; it then selects the best neighbour. Neighbour solutions of x using the first neighbouring structure are given in Table 5.6:

Table 5.6 – Neighbour solutions of x using the first neighboring structure

x_{11}	SA	AC	CB	BD	DT	$fitness = 14$
x_{12}	SC	CB	BD	DE	EF	FT	$fitness = 11$

Table 5.7 represents the neighbour solutions of x using the second neighbouring structure.

Table 5.7 – Neighbour solutions of x using the second neighbouring structure

x_{21}	SC	CB	BT	$fitness = 14$

The best neighbour of x using the first neighbourhood structure is x_{12}. The

algorithm will then move to x_{12} since its fitness is better than x. The same process is repeated over x_{12}. The best neighbor of x_{12} using the first neighborhood structure is the path $x^* = (SA, AC, CB, BD, DE, EF, FT)$ with length 10. The algorithm moves to x^* since its fitness is strictly better than x_{12}. The same process is repeated over x^*. However, x^* cannot be enhanced either by using the first neighboring structure or the second one. Therefore, the algorithm stops and return x^*.

Finally, the VNS method is based on two-neighbouring structures. At each time the proposed algorithm gets stuck at a local minimum, the structure of the neighbourhood is changed. By following this technique, the algorithm will exploit and explore wide regions of the search space. Adding more than two neighbouring structures will probably enhance the chance of the algorithm to find better solutions. However, that might increase the time to accomplish the preprocessing operation as well as the time to perform the VNS itself.

The strategy adopted to move from one solution to another is the best enhancing neighbour. Other techniques can be used such as the *first/best/least* enhancing neighbour. Such techniques may provide better results and extensive researches are conducting nowadays to decide which technique is the most efficient. To study the impact of such techniques on the performance of the VNS approach, we measure in Section 5.3 the quality of solutions obtained and the rate of convergence after varying the strategy to select the next solution from the neighbourhoods list of the VNS approach. In addition, in this work, the first neighbourhood structure is applied before the second one. In fact, deciding which neighbouring structure should be applied at which order has been always an issue for the VNS method. Further works will be done in near future to study if the order of neighbouring structure will affect the performance of the proposed approach, and eventually a random choice may be applied.

E Parent Selection

Generally speaking, the surviving probability of an individual is related to its efficiency in comparison with the other individuals in the population. In the evolutionary algorithms and like in the natural selection phenomenon, a stochastic character is introduced in the probability of selection.

The most famous stochastic techniques to accomplish the selection process are the roulette wheel and the rank selection. In the former, a roulette wheel is used where are placed all individuals in the population, the part size of each one depends on its fitness function. Therefore, the better the individuals, the higher the chances to be selected. The rank selection consists however, of ranking the population and then every individual receives a fitness from this ranking. The worst will have the fitness 1, the second worst 2, etc. and the best will have the fitness N (size of the

population). The roulette wheel is used in this work as a selection operator. For each individual, a selection probability is computed by dividing the WAR of the individual over the sum of fitnesses of all the individuals in the population.

I	GR	$P(xi)$
I_1	3,5	0,13
I_2	3	0,11
I_3	4	0,15
I_4	5	0,19
I_5	3	0,11
I_6	4	0,15
I_7	3,5	0,13

Table 5.8 – Example of evaluating individuals

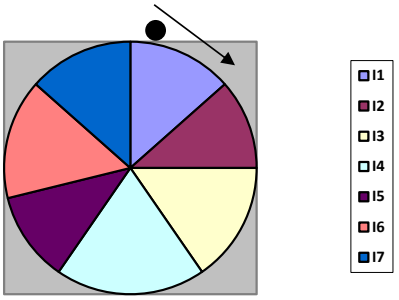


Figure 5.4 – Roulette wheel selection

Considering the example above (Table 5.8, Figure 5.4), the solutions have practically equivalent portions except for the individual I_4. This latter has the high probability selection since it is the fittest individual in the population. Therefore, its chance to survive and pass its genes to the next generation will be higher than the other individuals.

F Crossover

After the selection and during the reproduction operation, the individuals will be modified in order to generate new individuals (offspring) for the next generation. The two prominent operators to accomplish this operation are the crossover and the mutation.

The crossover consists of exchanging elements between two individuals initially selected in the goal of producing one or two new individuals. This operator is usually applied with certain probability (usually high) and may be accomplished in several ways depending on the structure of the problem itself.

When using the binary representation, the single and multiple point are the most used techniques to perform the crossover. In the former, one crossover point is selected, then the binary strings from the beginning of the chromosome to the crossover point are copied from one parent, and the rest is copied from the second parent. In the latter such as the two-point crossover, two crossover points are selected, then the binary strings from the beginning of the chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point are copied from the second parent and the rest is copied from the

first parent.

The single point crossover has been used as a crossover operator. Experimental results show that using more than one crossover point might increase the diversity of the proposed approach. Therefore, the algorithm's chance to find better solutions will also increase. However, that may be at the expense of additional computational efforts. It is decided therefore to only use the simple crossover technique. The crossover point is chosen to be any exchange node that is shared between parents. After selecting two individuals according to the selection operator, the new individuals (offsprings) are produced w.r.t the crossover point. By doing so, a new population having twice the size of the current population is produced. The best half individuals are then selected for the next generation and the rest are ignored. The goodness of an individual is measured according to its weighted average rank (WAR).

The higher the individual's WAR, the higher its chance to pass its genes to the next generation. It is also worth mentioning that the best solution at each generation is copied as it is to the next generation in order to avoid loosing the elite solution. Since there is a chance that the same individual is duplicated in the population as the generations go on, duplicated individuals are therefore replaced with newly generated individuals. This process will undoubtedly increase the diversity within the population. We present in Figure 5.5 an example to illustrate the crossover operation. Two parents P_1 and P_2 have been selected according to their fitness. The two offsprings O_1 and O_2 have been generated after exchanging parts between P_1 and P_2 w.r.t the crossover point which is the train platform three. Notice that there are only three candidate nodes for crossover in that example, the train platforms six, three and two.

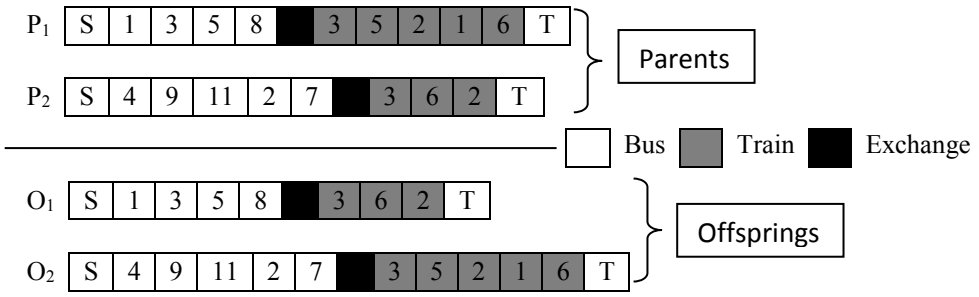


Figure 5.5 – Single point crossover

One point worth mentioning is that after the crossover operations used in this method, the algorithm does not care about the feasibility of the new generated path. The algorithm will always end up with a feasible path from the source to the destination. Therefore, the algorithm does not loose time to check the validity

of the offspring nor to perform some additional operations to repair the infeasible solutions.

G Mutation

The crossover operation may produce degenerate population. The algorithm may therefore get trapped in local minima. To overcome this issue, we perform the mutation operation.

The VNS has been used as a special mutation operator. That is, a VNS is applied over the population's individuals. By doing so, the algorithm is guided towards new regions within our solution space. The algorithm's chance to find better solutions will therefore increase.

Furthermore, applying VNS will ensure that the genetic algorithm maintains a sufficient diversity level that prevents premature convergence. The mutation operation has been applied at each time we perform the crossover. The conventional mutation operation used in GAs is usually applied with low probability. However, in the proposed hybrid approach, the mutation is always applied after crossover. The purpose of doing that is to allow the algorithm to avoid local minima by preventing the population of individuals from becoming too similar to each other, thus slowing or even stopping the evolution.

Other mutation techniques have been applied such as order changing between platforms, but it has been checked that the mutation becomes less efficient and it may provide invalid paths. An additional process should therefore be applied to reform infeasible paths. As a result, the mutation computational time will increase. It is decided thereby against using such traditional mutation techniques.

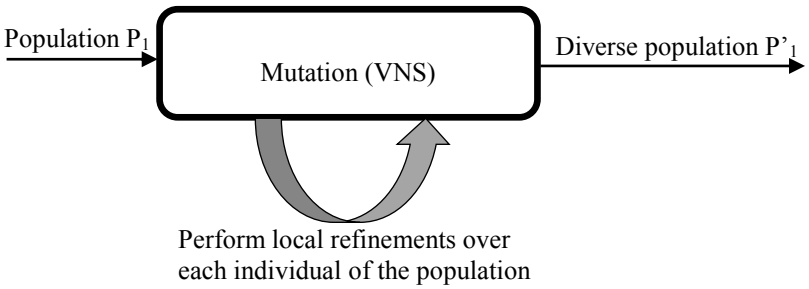


Figure 5.6 – Mutation scheme

In Figure 5.6, the scheme of the mutation operation is presented. As input, the VNS approach that is used to perform the mutation takes the population of

individuals. After performing the VNS on each individual of the population, the output of the mutation would be a set of better individuals in terms of quality and diversification level.

H Terminating Condition

Heuristic approaches do not guarantee finding the optimal Pareto front set. They do not therefore have the ability to automatically stop performing when a set of solutions is detected. Additional terminating conditions should then be introduced in order to allow the convergence of the algorithm.

A maximal number of generations, a fixed execution time, and no modifications in the population elements can be considered as algorithm stopping criteria. Two stopping criteria have been used in this study. Firstly, the algorithm stops when it fails to find interesting solutions during several continuous steps. An interesting solution is a new generated individual that is not dominated by any of the individuals in the current population P or it is a solution that at least dominates one individual in P.

100 generations are used as a number to ensure a fixed state in the population. Another stopping criterion is until the algorithms reaches the maximal number (500) of generations. It has been noticed after some experimentations that the algorithm visits a wide range of the search space rapidly. Thus, there is a big chance that the algorithm converges after few generations. That explains the small number of generations used as stopping criteria.

5.3 Experimental Results

We present in this Section the various experimentations that have been done in order to assess the performance of the proposed approaches.

A Experimental Setup

To assess the performance of the proposed work, we have developed an advanced web-based routing application using the real data of the French Region Île-de-France that includes the city of Paris and its suburbs. Data are provided by the transport organization authority that controls the Paris collective transport network and coordinates the different transport companies operating in Île-de-France, mainly the RATP, the SNCF and Optile.

Data comprise geographical information, as well as, timetable information for four transport modes (bus, metro, railway, and tram). More precisely, data encompass 17,950 stations; 41,047 platforms; 195,000 transfers; 303,000 trips and 6,800,000 events for one day.

For more information about the geographical perimeter of this study, we show in Figure 4.2 in Chapter 4 a screenshot taken from Google Maps API for the French Region of Ile-de-France that includes the city of Paris and its suburbs.

Before analysing results, it is worth clarifying that several models from the literature suffer from high computational effort when reading and compiling data. However, in the proposed modelling approach, the generation and integration of the different sub-networks take less than 12 seconds. This validates the efficiency of the proposed model, as well as, the appropriateness of the data structures used during the implementation phase.

For instance, an event node with its associated edges corresponding to waiting and boarding, and alighting are not represented as three separated entities during the implementation. In contrast, they do belong to the same object. Information are then added inside one object to refer to that information.

Thanks to such implementation tricks, we succeeded at reducing the computational space to store all the model's components. Therefore, the size of the network resulting from the modelling phase does not constitute a bottleneck for the proposed approach to be integrated in real world routing system.

The following parameters have been used in the proposed GA-VNS and the pure GA: the initial population size is 5; the probability of crossover is 0.9; the mutation rate is 0.9; the population size is 5; the maximum number of generation is 500; the number of generations used to ensure a fixed state in the population is 100. The choice of these values is explained later in subsections.

Algorithms have been tested on an Intel core I5 machine of 8 GB RAM and Java is used as a programming language. Efficient and appropriate data structures have been used to guarantee fast access to information when needed and thereby improve the performance of the proposed approach.

Since the increased time to solve route planning issues decreases the utility of the journey planning system, the computational effort of any routing algorithm constitutes a critical success factor for its integration within an online routing system. Therefore, in this work, the evaluation of the proposed hybrid approach GA-VNS is done by comparing its running time with other approaches such as the algorithm of Dijkstra, a pure genetic algorithm and a pure VNS approach.

The algorithm of Dijkstra has been used as a reference for comparing other

approaches (GA,VNS, GA-VNS) since it provides the optimal set of nondominated solutions to go from one station at a certain departure time to another station. Although the bidirectional Dijkstra algorithm may be seen as an advanced way to speed up the standard Dijkstra algorithm, however, it cannot be used in this work since the exact arrival time at the destination node is not known a priori. Therefore, the standard algorithm of Dijkstra has been selected for evaluations.

The pure GA applied has the same scheme of the proposed GA-VNS except that VNS is not applied neither to enhance initial solutions, nor to perform the mutation operation. Rather, a standard mutation that consists of taking a subpath from an individual and replacing it with a newly generated path using the algorithm to generate initial solutions. The mutation in this case is applied with low probability of 0.1. Regarding the VNS approach, the moving strategy used is the best enhancing neighbour since this strategy is the most efficient one for the studied problem.

Moreover, the starting solution for the VNS is chosen arbitrary. It is worth to mention that the starting solution may affect the performance of the VNS approach. After empirically testing many starting solutions, it has been found that the performance of the VNS for the studied problem is not largely impacted by the starting point. Therefore, in this work, a random starting solution has been given as an input solution for the VNS.

Furthermore, since the proposed GA-VNS approach is based on two heuristic approaches, there is no guarantee to find the exact Pareto front set. Therefore, another key factor to assess the performance of one heuristic approach is to measure the quality of solutions found with respect to optimal solutions.

To do so, we compute the average distance between all solutions and a reference point and the same is done for the optimal method; the differential distance is then used as a gap metric. The exact approach is based on the classical shortest path algorithm of Dijkstra, which has been generalized to compute the exact set of non-dominated solutions to go from one station to another according to some predefined criteria. It is obvious that using this technique will only assess the average closeness of each solution set to the true Pareto Front. Other important points should be studied to assess an approximate Pareto set such as how well approximate solutions cover the whole extension of the true Pareto front; and whether the solutions in the approximation set are well spread and spaced among each other. We devote subsection C to study other importance performance indicators such as Epsilon, Spread and Hypervolume.

Finally, comparing the running time and the solutions quality of the proposed GA-VNS has been done by solving 10,000 routing queries and w.r.t the algorithm of Dijkstra, a pure GA and a pure VNS. Each routing request consists of a source node, a target node, and a departure time. Those parameters are uniformly generated at random. It is assumed in this work that requests allow all transport modes.

Although the proposed routing method is not limited to specific criteria, experimentations are done w.r.t the following criteria: the travel time, the cost and the number of transfers and the total walking time. One can add more criteria such as distance or CO_2 emission by charging their values on corresponding edges.

We present in Figure 5.7 the details of the experimental process. As can be seen, as input, a graph instance corresponding to the real transport network of the French Region Ile-de-France has to be built. The modelling approach explained earlier in this manuscript is used for constructing the final multimodal graph. Besides, a routing request is required for assessing the proposed approaches. Random as well as real world routing requests can be used for this purpose.

All tested algorithms have the same input, and for each of them several comparison criteria are then computed in order to decide later which approach is the most performant one for the emerging routing problem.

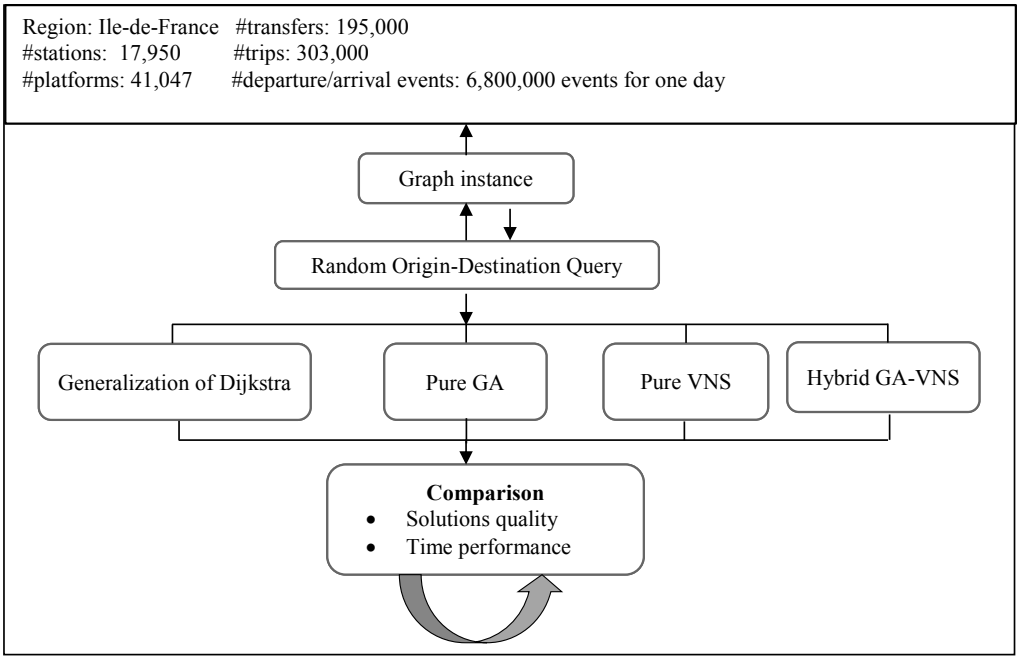


Figure 5.7 – Experimental setup for multicriteria routing using metaheuristics

B Experimental analysis

Before comparing the different approaches, it is worth mentioning that the performance of the VNS approach may be affected by the choice of the strategy adopted to move from one solution to another. Several strategies can be used such as the first, best or least enhancing solution among all solutions belonging to the set of

neighborhood list. Theoretically speaking, deciding which strategy is the most efficient is a very difficult task since it largely depends on the problem itself. For this seek, we have studied the impact of several moving strategies.

Results in Table 5.9 show that the moving strategy has an impact on the performance of the VNS approach regardless the number of criteria taken into account. While the first enhancing strategy may help to reduce the computational time of the VNS approach, it may decrease the quality of solutions.

Furthermore, when using the worst enhancing strategy, we show the quality of solutions is enhanced in comparison with the first enhancing technique and the results are very close to the best enhancing strategy. However, that improvement will be at the expense of additional computational time. Finally, using the best enhancing outperforms the other two strategies in terms of the quality of solutions found. However, it takes more time than the first improving strategy. Therefore, we decided to use the strategy that gives the best solutions since the additional time required to get the best enhancing neighbour is not very limited.

Table 5.9 – Impact of different moving strategies on the performance of VNS

Criteria	△	**FE**		**LE**		**BE**	
		◊	□	◊	□	◊	□
f_1	1.6	51	5.88	58	4.44	55	4.51
f_2	1.5	52	6.64	61	4.14	53	4.16
f_3	1.52	51	7.14	64	5.36	51	5.31
f_4	1.54	55	6.83	57	5.68	56	5.66
(f_1, f_2)	2.18	75	12.23	93	8.25	81	8.15
(f_1, f_2, f_3)	20.60	84	13.54	101	10.29	92	9.16
(f_1, f_2, f_3, f_4)	190.17	99	15.45	124	13.36	110	12.90

f_1: travel time f_2: monetary cost f_3: number of transfers f_4: total walking time
△: running time of the exact approach of Dijkstra **in seconds**
FE: First Enhancing LE: Least Enhancing BE: Best Enhancing
◊: average running time **in milliseconds** □ average gap to the optimality in %

After empirically identifying the most efficient moving strategy for the VNS approach, we start evaluating the proposed approach by limiting the routing query to only one criteria (*i.e.* we start by travel time). That is, we want to ensure that the proposed model allows computing multimodal routes in a single criteria context and the proposed algorithm provides correct results. Analysing and comparing results with real world routing applications have verified the ability of the proposed model in providing correct itineraries when applying the routing algorithm.

When it comes to assessing the computational performance, the results in Table 5.10 indicate that the hybrid GA-VNS does not exceed the ratio of 70 milliseconds

to handle a routing request. Conversely, the running time of the exact approach of Dijkstra increases to 1,600 milliseconds. Thus, the acceleration column of the proposed GA-VNS, which is computed by dividing the running time of the algorithm of Dijkstra over the running time of the proposed GA-VNS, indicates a maximal gain of 2.42.

While the exact approach guarantees finding the exact Pareto front set, the average gap to the optimality of the proposed GA-VNS in a single criterion does not exceed the ratio of 3%. Results in Table 5.10 also show that the running time of the other heuristic approaches GA and VNS is better than the algorithm of Dijkstra and the hybrid GA-VNS. This can be explained by the fact that such approaches rapidly converges in comparison with the proposed GA-VNS. The rapid convergence of the pure GA and VNS can be also noticed from their average gap w.r.t the hybrid GA-VNS. While the gap of the pure GA increases to 3.45, the gap of the pure VNS may reach 5.66.

By analyzing the result below, it can be said that the proposed hybrid GA-VNS provides better solutions than the pure GA and VNS. However, that will be at the expense of additional computational time. It can also noticed that using a heuristic approach to solve the single criterion shortest path is not very attractive since the exact algorithm of Dijkstra is efficient enough to provide optimal solutions within reasonable computational time (no more than 2 seconds). Finally, the results in Table 5.10 also indicate that when changing the criterion, the algorithm's search space does not relatively change and so does the ultimate running time.

Table 5.10 – Single criterion experimentations

Criteria	△	**GA-VNS**			**GA**		**VNS**	
		◊	□	⋆	◊	□	◊	□
f_1	1600	70	2.88	2.28	62	3.19	55	4.51
f_2	1550	64	2.58	2.42	58	3.25	53	4.16
f_3	1520	63	2.69	2.41	57	3.45	51	5.31
f_4	1540	66	2.43	2.33	61	3.16	56	5.66

f_1: travel time f_2: monetary cost f_3: number of transfers f_4: total walking time
△: running time of the exact approach of Dijkstra **in milliseconds**
◊: average running time **in milliseconds** □ average gap to the optimality in %
⋆: average acceleration compared to Dijkstra

After studying the performance of the proposed approaches in a single criterion environment, we now analyse their behaviours in a multicriteria context. While using the exact approach to compute routes in a single criteria context was done without any limitations, computing Pareto routes turns out to be more challenging.

Results in Table 5.11 indicate that the average running time of the exact ap-

proach in two criteria context may reach 2.18 seconds. It, however, increases to 20.60 seconds when considering three criteria and to 190.17 seconds when dealing with four criteria. As can be easily noticed, using the exact approach to support online users' queries is not satisfying in a real context due to its high computational time.

It can be also noticed that the average running time of the exact approach exponentially varies with the number of criteria. More criteria lead to an exponential increase in the search space an thus, an important increase in the running time. When it comes to assessing the performance of the proposed heuristic approaches, the results indicate that their average running is highly better than the exact approach.

While answering a routing request may take more than three minutes when using the exact approach and considering four criteria, the heuristic approaches provide answers within no more than 170 milliseconds. It can be also seen from Table 5.11 that the running time of the pure VNS is the best one. However, that decreases the quality of its provided solutions. While the average gap to the optimality of the proposed GA-VNS reaches 3.24 %, the average gap of the pure GA increases to 7.46 % and 12.90 % for the pure VNS.

By analysing those results, we can notice that the proposed hybrid GA-VNS represents the best compromise in terms of running time and the quality of solutions provided. Consequently, our algorithm is efficient enough to replace other approaches and to be used in real world routing system.

Table 5.11 – Multiple criterion experimentations

Criteria	△	**GA-VNS**			**GA**		**VNS**	
		◊	□	⋆	◊	□	◊	□
(f_1, f_2)	2.18	0.11	3.14	20	0.087	4.25	0.081	8.15
(f_1, f_2, f_3)	20.60	0.14	3.19	147	0.095	5.90	0.092	9.16
(f_1, f_2, f_3, f_4)	190.17	0.17	3.24	1118	0.120	7.46	0.11	12.90

f_1: travel time f_2: cost f_3: number of transfers f_4: total walking time
△: running time of the exact approach of Dijkstra in **seconds**
◊: average running time **in seconds** □ average gap to the optimality in %
⋆: average acceleration compared to Dijkstra

C Performance Indicators

In single criteria optimization the evaluation of algorithms is easy since algorithms have one value for one optimal solution. However, this is not the case in MOSP where each solution has k values for k objectives, and each algorithm generates trade-off solutions that constitute an approximation of the Pareto Front. Thus to

evaluate two algorithms, we have to compare two sets of solutions generated by each algorithm. When all the solutions generated by an algorithm B are dominated by a solution of algorithm A, in this special case we can say that A is better than B. However, it is not easy to compare two set of solutions where some solutions of A dominate some solution of B and viceversa.

Therefore, there are special performance indicators that can be used to evaluate an approximation of the Pareto front. Most of these performance indicators evaluate the solutions on two criteria: the convergence (*i.e.* closeness to the optimal Pareto set) and the diversity (*i.e.* spread of solutions on the Pareto front). There are many quality indicators that are proposed in the literature such as Hypervolume, Epsilon, Spread, etc. Each quality indicator is intended to measure one or both of the criteria at the same time as show in Table 5.12. In another perspective, quality indicators can be classified into two main categories: unary indicators that assign a score for a set of solutions, and binary indicators that compare two sets of solutions and assign a score to a set relatively to the other one.

Table 5.12 – Quality indicators with regard to convergence and diversity

Measure	**Epsilon**	**Spread**	**Hypervolume**
Convergence	X		X
Diversity		X	X

C.1 Epsilon

The epsilon indicator has been presented in [**?**]. This indicator has two versions: multiplicative and additive. In its additive version, the epsilon indicator is used to calculate the minimal translation factor ε that should be added to every solution of an approximation set A in order to dominate the solutions of the Pareto front PF*. This indicator can be defined as follows:

$$I^1_{\varepsilon+}(A) = inf_{\varepsilon\in\mathbb{R}} \left\{ \forall z^2 \in PF^* \exists z^1 \in A : z^1 \prec \varepsilon^{Z^2} \right\} \tag{5.5}$$

where $z^1 \prec \varepsilon^{Z^2}$ if and only if $\forall 1 \leq i \leq n : z^1_i < \varepsilon + z^2_i$ for minimization problem.

The smaller the value of ε, the better the approximation. It is good to note that a normalization of the objective functions is required in order to obtain efficient results when the scale is not the same for all the objective functions.

C.2 Spread

The spread indicator measures the spread level of the solutions that belong to an approximation set according to the Pareto front [**?**]. It is defined as below:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \tag{5.6}$$

where d_i is the distance between two adjacent solutions, $\bar{d}$ is the average value of these distances, d_f and d_l are the Euclidean distances between the extreme solutions of the Pareto front and the approximation set. A spread value of zero refers to an ideal distribution, then the smaller the value the better the approximation.

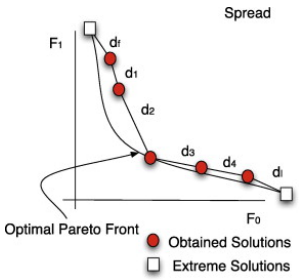


Figure 5.8 – Distances from extreme solutions

C.3 Hypervolume (HV)

The hypervolume indicator computes the volume in the objective space of the space covered by the set of the nondominated solutions A of an approximation set [**?**]. This indicator is very popular in the literature since it reflects the quality of an approximation based on two criteria: diversity and closeness to the true Pareto front. In its unary form, the hypervolume is calculated based on a reference point Z_{ref} (*anti-optimal* or *nadir* point) which refers to the worst possible point in the objective space as shown in Figure 5.9 where we deal with a minimization problem. A hypercube v_i is formed of each solution $s_i \in A$ and the reference point Z_{ref}, as diagonal corners. The larger the hypervolume, the better the set of solutions. The hypervolume is the union of all hypercubes:

$$HV = volume\left(\bigcup_{i=1}^{|A|} v_i\right) \tag{5.7}$$

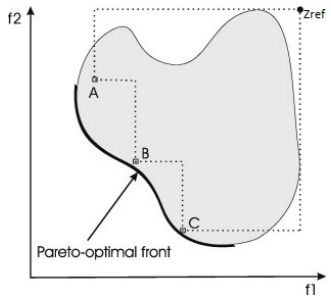


Figure 5.9 – Hypervolume indicator (HV)

D Performance assessment

In order to have a better evaluation of the different approaches that have been used for solving the multicriteria shortest path problem, the performance indicators introduced above are computed for each method.

Results in Table 5.13 show that the proposed GA-VNS has the lowest Epsilon value in comparison with both the Genetic Agorithm and the Variable Neighbourhood Search. This means that the solutions provided by the Memetic Algorithm are closer to the real Pareto front in comparison with other approaches.

Assessing the hypervolume indicator proves also that the nondominated solutions obtained from the memetic approach are well distributed in comparison with those solutions obtained from other approaches but the pure Spread indicator shows a short advantage to VNS.

Table 5.13 – Performance assessment of the approximation sets of solutions

Criteria	**GA-VNS**			**GA**			**VNS**		
	E	**S**	**HV**	**E**	**S**	**HV**	**E**	**S**	**HV**
(f_1, f_2)	0.021	0.541	0.693	0.071	0.662	0.414	0.085	0.516	0.341
(f_1, f_2, f_3)	0.027	0.549	0.685	0.098	0.693	0.395	0.105	0.521	0.311
(f_1, f_2, f_3, f_4)	0.039	0.553	0.661	0.11	0.712	0.388	0.139	0.558	0.294

f_1: travel time f_2: cost f_3: number of transfers f_4: total walking time
E: Epsilon (smaller better) **S:** Spread (smaller better) **HV:** Hypervolume (larger better)

Finally, another indicator to compare the multicriteria approaches is the number of nondominated solutions found at the end of the run. That is, the size of the final archive is an important aspect to study when applying several approximate approaches to solve a multicriteria problem. Results in Table 5.14 show that the number of nondominated solutions resulting from the memetic approach is better than other approaches.

Table 5.14 – Number of resulting nondominated solutions in the final archive

Criteria	GA-VNS			GA			VNS		
	$>$	$<$	˜	$>$	$<$	˜	$>$	$<$	˜
(f_1, f_2)	3	69	35	6	39	24	4	29	12
(f_1, f_2, f_3)	4	96	51	9	78	31	3	36	19
(f_1, f_2, f_3, f_4)	6	130	75	8	84	46	6	54	27

f_1: travel time f_2: cost f_3: number of transfers f_4: total walking time
Number of solutions: $>$ Minimal $<$ Maximal ˜ Average

E Tuning Parameters

As in most of metaheuristics, tuning the parameters is one of the hardest part while accomplishing the implementation phase. Unfortunately, there is no standard parametrization rule since parameters highly depend on the structure of the optimization problem itself.

Several parameters have to be tuned for the proposed algorithms in order to achieve the best performance. We start with the initial population size. As previously mentioned, we limited the initial population size to five. One reason for that is that, the algorithm used for generating initial feasible solutions is not always capable of providing as much solutions as we want. Another reason originates from the experimentations.

Experimental results show that having less than five initial solutions will lead to premature convergence. In other words, the initial population becomes less diverse. Consequently, the algorithm usually converges to poor local minima very rapidly.

On the other hand, having more than five initial individuals may increase the diversity in the first population. However, that will be an expense of additional computational time and in most cases to the same quality of final computed solutions. Thus, we have fixed the initial population size to five.

We have also analysed the crossover and mutation rates. As in most traditional GA schemes, the crossover rate is always high, while the mutation probability is very low. However, in our case the mutation is always applied after the crossover. Experimental results have shown that the best crossover and mutation rates for our problem is 0.9. Since the crossover is a convergence operation, which is intended to pull the population towards a local minimum/maximum, performing the crossover over all individual will lead to premature convergence. On the other side, results indicate that using less than 0.9 as a crossover probability will affect the final quality of the solutions. That is, the average gap to the optimality will increase as the crossover rate decreases.

Results have also indicated that decreasing the mutation rate will prevent the algorithm from converging towards interesting regions within the search space. This can be explained by the fact the mutation is a divergence operation and is usually intended to occasionally break one or more members of a population out of a local minimum/maximum space and potentially discover a better minimum/maximum space. On the other hand, decreasing the mutation rate to less than 0.9 will lead to premature convergence. Thus, the quality of final solutions found will be degraded.

Therefore, parameters used to tune the proposed GA have been experimentally found. Using such parameters will lead to achieving the best performance and thus increasing the efficiency of the introduced GA-VNS. It is worth mentioning that other values can be used to increase the convergence rate of the algorithm, but without guarantying better solutions.

5.4 Synthesis

The Multicriteria Shortest Path problem was solved in this chapter using a Memetic Algorithm. This latter lies in a combination between Genetic Algorithm and Variable Neighbourhood Search. The algorithm starts with building initial feasible solutions using a constructive double search algorithm. The quality of the initial population is then enhanced by performing a VNS over each individual. Besides, genetic operators and mainly selection, crossover, mutation and replacement are applied until meeting one of the stopping criteria. The crossover is a single point based on a random common node between the parents. The selection is a roulette wheel based on WAR process. The replacement keeps the best one and then the best among the union of parents and offsprings. The VNS is used as a mutation operator. A vector of integers is used to encode a solution and an n-dimensional vector is used for the evaluation. An archive is also used to store the set of nondominated solutions during the evolution. To perform the VNS, two neighbourhood structures have been designed. The algorithm starts with the first neighbouring structure and switches to the second one whenever a local minimum is detected.

To assess the performance of the proposed algorithms, we have solved real world itinerary problems defined on the transport network of the French Region Ile-de-France. Results have indicated that the proposed Memetic Algorithm is capable of providing high quality solutions in a reasonable amount of computational time. The algorithm benefits from the efficiency of GAs in exploring a large search space, and takes the advantages of VNS in efficiently exploiting promising search regions.

Chapter 6

Stochastic Route Planning

The aim of this chapter is to study several shortest path problems in a stochastic and time dependent multimodal network. We will start by describing some important aspects that may arise in a stochastic system such as the definition of the randomness associated with components, and the description of their level of stochasticity. We then switch to the distribution function of a random variable. After that, a monocriteria routing algorithm is developed in order to solve the least expected shortest path problem. Furthermore, a multicriteria routing algorithm is introduced whereby several robust alternatives are provided to go from an origin place to a destination. Experimentations are then provided, and the last section is devoted to outline this chapter.

Sommaire

Related publications:

1. Omar Dib, Alexandre Caminada, Marie-Ange Manier, and Laurent Moalic. A memetic algorithm for computing multicriteria shortest paths in stochastic multimodal networks. The Genetic and Evolutionary Computation Conference GECCO. July 15-19, Berlin, 2017. (Qualis: A1, ERA: A).
2. Poster on Stochastic Route Planning Download
3. Stochastic Route Planning Watch Video

6.1 Introduction

We have proposed in previous chapters an advanced modelling approach as well as, several routing algorithms that solve basic and advanced routing issues. We indicated also that such algorithms can handle the dynamic aspect of the transport system. The dynamic term has been used to refer to the fact that some edges highly depend on the time they are crossed. Such dependency is usually known a priori. In another term, a dynamic edge has been considered as a set of static edges. Each static edge is linked to a departure time interval. For instance, in a road network, the same edge can take several weights depending on the day and time. At 7:00 am, the travel time to cross a route segment is usually high in comparison with midnight. In contrast, at 2:00 am, the travel time is usually low.

It is obvious that handling time dependency is crucial to provide passengers with correct and precise itineraries. However, that is not sufficient due to the stochastic variations that the transport system may encounter. Such a stochastic property may be caused by several unanticipated events such as accidents, weather conditions, drivers' behaviours, closure of roads, cancelling train or metros.

Omitting stochastic variations inside routing algorithms will provide routes without any guarantee about their quality whenever an anticipated event occurs. In many cases, the optimal solution of an optimization problem may become very bad if a small variation occurs to one or several decisions variables. For instance, a study we have conducted has shown that the shortest path between two stations has a travel time of 30 minutes; the travel time of the same route increases to 50 minutes after forcing a vehicle to depart one second less than its theoretical schedule. This validates the fact that small variations inside even one variable of a complex system may highly reduce the quality of the solutions provided.

One approach to tackle unanticipated events is to solve again the optimization problem after updating the state of all variables and constraints in the optimization problem. For instance, in the transport system, one can easily reapply the routing algorithm over the updated multimodal graph. Modifications in such context may become in the form of changing departure time, cancelling vehicles, adding intermediate stops, updating entities' properties or even removing components from the graph, etc. It is obvious that such a method requires having the possibility of updating the model. While some models allow that, updating some others may be not feasible due to the high computational time needed for modifications or due to the nature of the model itself.

Furthermore, solving again several times the optimization problem may not be an easy task even if the update of the model is easy. In some optimization problems, the execution time is very high. Thus, a real time interaction with the optimization module is not possible. Additionally, in some optimization problem, the solution

itself may not be changed whenever a situation occurs due to the cost required for implementing the solutions (*e.g.* implementing electrical vehicles stations).

Besides, readjusting a solution after the occurrence of an event may usually provide poor solutions although an optimal shortest path algorithm is applied. Indeed, the optimal solution found after an anticipated event occurs may be dominated by a robust solution computed before the occurrence of the event.

As a result of the above mentioned reasons, the reactive method that consists of reacting to a situation after the occurrence of unanticipated event is not the optimal way to deal with stochastic problems.

To overcome that, another approach can be used, which consists of providing routes, whose quality does not drastically change against random variations, but several difficulties arise when using that method to tackle the stochasticity.

In next sections, we present several aspects of randomness inside the components of a complex transport system and some distribution functions. After that we will introduce the algorithms that were developed for stochastic optimization.

6.2 Stochastic Components

Before solving any optimization problem in a stochastic environment, it is crucial to identify the components in which the randomness is taking place. This will usually help when the optimization model is built and resolved.

Indeed, in many real world systems, the uncertainty may not be found in all components. Some of these latter are highly susceptible for random variations, while others do not depend on other factors.

For instance, the square root function $\sqrt{x}$ will always give the same result for the input x. Thus, this function does not depend on other random variables. In contrast, in the function $\sqrt{x + rand(0,10)}$, the component $rand(0,10)$ will always generate a random variable between 0 and 10. As a result, the final result of any value of x will vary depending on the value of the random component $rand(0,10)$.

When it comes to the transportation system, several components in the multimodal graph can be qualified as stochastic. For instance, in the pedestrian network, the travel time to go from one place to another one depends on many external random factors such as the speed of each person or the weather conditions. Thus, any edge belonging to the pedestrian network can be qualified as a stochastic component.

Additionally, transferring from one transport mode to another one can be af-

fected by the profile of the passenger (*e.g.* local, tourist), the weather conditions or the users behaviours. Thus, the transfer edges can also be seen as stochastic components.

Moreover, edges in the road network can also be stochastic due to the unanticipated drivers' behaviours, the weather conditions or the traffic jam.

Besides, in a public transport network, the departure of vehicles is also a stochastic component since it can be affected by random factors such as the number of passengers waiting on platforms, the behaviours of vehicles' drivers or any technical problems.

It can therefore be remarked that most components in the final multimodal graph are stochastic since they all have a stochastic travel time parameter, which may depend on many random factors. In addition, such components are not only stochastic, but their random variations can also depend on time. That means that the random variation of a travel time metric over an edge can depend on the time at which the edge is crossed. This dependency leads to a stochastic time dependent multimodal network.

Having time dependent stochastic components will cause additional efforts to analyse such components and to compute itineraries.

6.3 Degree of Uncertainty

It has been indicated in the previous section that the stochastic travel time makes most components in the multimodal graph stochastic. Considering then that stochastic aspect will be very costly in terms of running time and memory space, then in most real world applications, assumptions are made so that randomness is considered on few components.

Determining in which components the uncertainty has to be handled is a very difficult task. Indeed, in a complex system, small variations in any component may induce drastic consequences. However, in most real world applications, decision variables are carefully analysed and stochasticity is only considered on variables, whose variability highly degrades the quality of the solution.

In the proposed modelling approach, the degree of randomness of each component is analysed. It can be noticed that walking in the pedestrian network is not often affected by many external factors in comparison with transferring from one transport mode to another.

It can also be remarked that going from one station to another will not be

affected by a lot of factors. As a result, stochasticity on those components can be neglected, and only time dependency is considered.

In contrast, when it comes to the departure time of vehicles, it is obvious here that considering the uncertainty is very important. That is, some assumptions have to be made to reduce the number of stochastic components, and thus, reduce the time and memory space complexity on that topic.

6.4 Distribution Function

After identifying the stochastic components that have to be considered, studying their variation has to be done. Analysing how a random variable may change refers to proposing a function that predicts its value in the future with respect to some parameters.

To predict the state of a system, several approaches can be used. For instance, many simulations can be accomplished, and values related to all components are then registered and used to build a probabilistic model. Nowadays, other alternatives can be used such as machine learning and big data science, which usually exploit historical and real time information to anticipate the state of a system in the future. Deciding which prediction method is the best is a hard question since it depends on many parameters such as the computational time requirements, the precision requirements and the available data. It is worth mentioning that in this thesis we do not focus on the prediction methods, predictions are rather given as an input for our work. Indeed, the prediction module is the main subject of another PhD thesis [**?**] in the same project.

The result of any prediction method will come in the form of a distribution function that predicts the state of decision variables with a certain probability. A distribution may come in the form of a random function or well defined one such as gaussian, uniform or exponential distributions.

In this chapter, the variation of the travel time metric, as well as the departure time of vehicles, is assumed to vary according to a normal distribution function with predefined mean μ and standard deviation σ.

We show in Figure 6.1, an example of a normal distribution function with $\mu = 60$ seconds and $\sigma = 10$. The variation refers to the travel time of a transfer edge. It can be seen from the figure that the probability of accomplishing the transfer is mainly concentrated around 60 seconds.

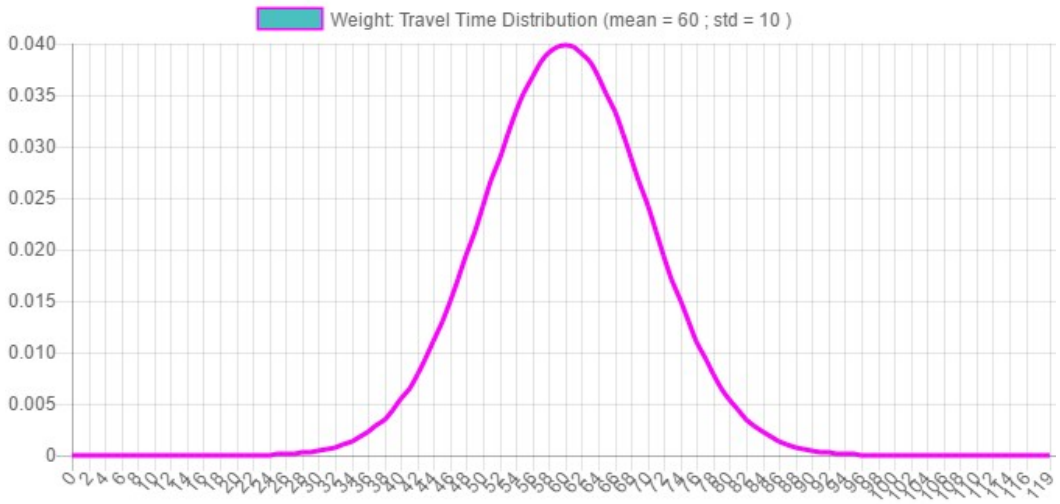


Figure 6.1 – Normal distribution for travel time metric

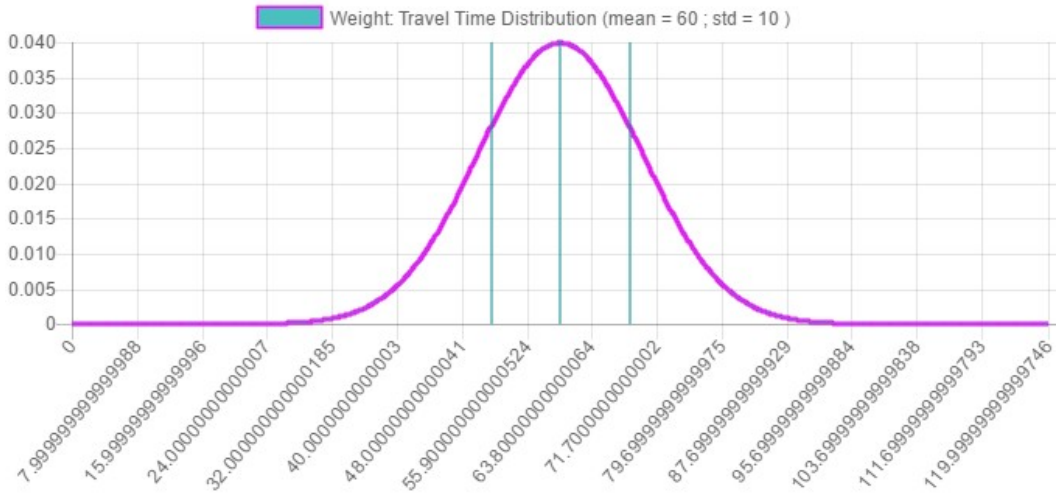


Figure 6.2 – Discretized normal distribution

Indeed, in a normal distribution function, several parameters have to be tuned such as the lower and upper bounds, the mean and the standard deviation. In this work, we will assume that the mean is the theoretical travel time obtained from the theoretical timetables information. The lower bound is set to the value of the mean minus five minutes and the same is applied for the upper bound. The standard deviation is set to 10. Those values can be changed according to the preference of users or transport operators.

To integrate stochasticity inside routing algorithms, a discretized version of the normal distribution function is utilized. Each discrete point contains a value of

travel time and a probability of occurrence. We show in Figure 6.2, an example of a discretized function. Three discrete points are considered.

It is worth mentioning that each discrete point refers to a scenario. That is, a specific state of the transport system is associated with each discrete point. Proposing then robust itineraries will be done with respect to all of these scenarios.

Besides, it has been noticed that in the transportation system, vehicles are usually prone to delays. Thus, the travel time will usually be higher than it has been expected. Therefore, the distance between the mean value and the lower bound will be less than the distance between the mean value and its upper bound. This reflects better the real world scenarios.

We present in Figure 6.3, a distribution function of a travel time metric that consists of going from one platform to another using a vehicle. The mean value is equal to 460 seconds when the standard deviation is set to 20. It can be seen from the figure that the lower bound is equal to $(\mu - 60)$ where the upper bound is equal to $(\mu + 340)$. Thus more chances are given to delays than to earlier arrival. Also, that can be adjusted by increasing the number of considered scenarios in the part that reflect delays in the distribution function.

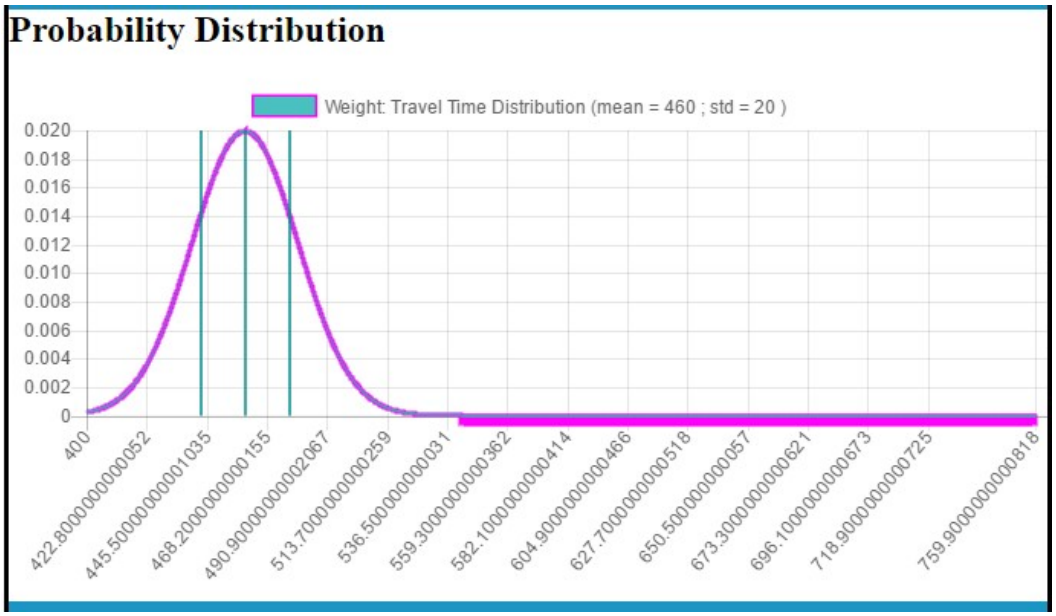


Figure 6.3 – Asymetric discretized normal distribution

We show in Figure 6.4 the cumulative distribution function of the travel time stochastic variable. Three discrete points are taken into consideration. The figure shows that the maximum travel time along the edge is 760 seconds. In average, the travel time is equal to 460 seconds.

The cumulative distribution is another way to represent the variation of a stochastic variable. It gives the probability that the stochastic variable takes a value less than a number. This can be useful when providing passengers with itineraries. Most passengers can be interested in knowing the maximal arrival time before which they will arrive to their destination.

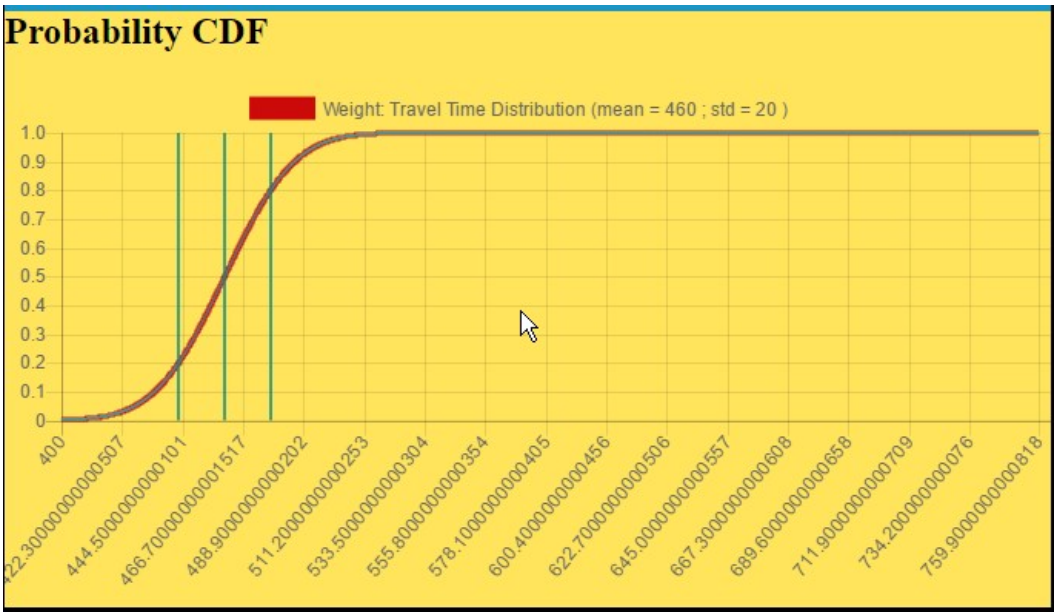


Figure 6.4 – Cumulative distribution function for travel time metric

6.5 Monocriteria Stochastic Algorithm

Several routing issues can be solved in a time variant stochastic multimodal network. In real world situations, the passengers usually ask for the least expected shortest path. Providing the travel time distribution of this path can also be very important for a wide range of passengers.

We already explained in chapter 3 how stochastic edges can be dealt with in the proposed representation. We introduce in this section the routing algorithm that has been proposed to solve the least expected shortest path problem in a multimodal network where edges' weights can take random values.

As can be noticed, minimizing the stochastic travel time is the only metric considered in this monocriteria section. The formula related to the computation of this objective is represented as follows:

$$Z(path) = \beta(path) = \varpi(D) \tag{6.1}$$

where D represents the node corresponding to the arrival station, the objective β to minimize represents the average (expected) arrival time of the journey and ϖ is the function responsible for computing the average arrival time at a given node. ϖ takes a time distribution function and returns its average value. It is worth mentioning that β is similar to the objective f_1 described in Equation 3.7 in Chapter 3 but in a stochastic context.

The proposed algorithm belongs to the dynamic programming category of algorithms and can be seen as a variation of the algorithm of Dijkstra. As input, the algorithm takes a routing request consisting of a departure station ds, an arrival station as, a departure time dt, and a multimodal graph. The output of the algorithm is the least expected shortest path that goes from ds to as respecting dt. As most routing algorithms, the proposed algorithm has three main phases: the initialization, the relaxation, and the visualization.

A Initialization Phase

In the initialization phase, a priority queue pq is initialized. pq will contain objects of type *Solution*. This latter is a data structure containing information about a path between the departure station ds and a platform belonging to another station. More precisely a solution s of type *Solution* contains the following information:

- The identifier of its parent platform p
- The travel time distribution from ds to p
- The identifier of the edge that leads to the creation of this solution
- The identifier of the previous solution that leads to the creation of this solution

The last two variables are used when reconstructing the final path in order to make it visual for the user.

It is important to mention here that a solution has a travel time distribution and not a single travel time value such as in the static case in previous chapters. More precisely, the travel time associated with a solution may vary within certain interval with respect to a probability law.

The travel time distribution is implemented as a separate entity that contains a list of discrete points dp, where each dp has an arrival time associated with an arrival probability. Based on this vector of discrete points, one can compute the expected arrival time, which represents the mean or the expectation of the set of discrete points. Furthermore, other parameters can be computed such as the

standard deviation, the variance, the minimum arrival time to the platform and the maximum arrival time at which a passenger will arrive at the corresponding platform.

After initializing pq, each platform node p is assigned a solution s that has the following properties:

- The identifier of the parent platform of s is set to the identifier of p.
- The travel time distribution is initialized with one discrete point with the probability p_i equal to one and ∞ as an arrival time at. Such two values are set to reflect the fact that at the beginning of the algorithm, reaching any station in the network from the departure station ds is possible ($p_i = 1$) but the travel time will be very high. Modifying the travel time distribution will be explained later in this section.
- The identifier of the edge that leads to the creation of this solution is set to -1 and so we do for the identifier of the predecessor solution.

Since the algorithm has to respect the predefined departure time dt, solutions belonging to the platforms of the departure station ds are initialized differently as follows:

- The identifier of the parent platform of s is set to the identifier of p.
- The travel time distribution is initialized with one discrete point with the probability p_i equals to one and the departure time dt added to the time required to access the platform from the arrival time to its parent station node as. We consider in this case that edges linking a station with its platforms take static values. The uncertainty on these edges is therefore disregarded because it is usually small w.r.t. the full journey and quite straightforward.
- The identifier of the edge that leads to the creation of this solution is set to the edge that links the platform with its parent station.
- The predecessor solution is set to $null$. This value will help latter to detect the departure station while reconstructing the final found path from the arrival station.

The last step in the initialization phase is to add the platforms belonging the departure station to the priority queue. After that, the algorithm will accomplish some operations in order to propagate the travel time distribution among other platforms containing the platforms belonging to the arrival station.

B Relaxation Phase

The body of the algorithm is called the relaxation phase. The main operation in this phase consists of pulling the highest priority element (*i.e.* solution) s from the priority queue and then relax the adjacent edges of the parent platform of s.

A solution s_1 is higher in priority than another solution s_2 if the mean of the travel time distribution of s_1 is smaller than the mean of the travel time distribution of s_2. By choosing the mean to compare the solutions, we ensure that at each iteration, the nearest expected platform from the departure station ds will be relaxed from the queue.

After pulling the highest solution s from pq, the relaxation of the adjacent edges of the parent platform p of s is done. Relaxing an edge e consists of building a candidate solution cs based on s and the characteristics of e (type, travel time distribution...). Additionally, in the relaxing phase, the best ever found solution stored on the head platform to is updated if the quality of the current candidate solution cs is better in terms of the criterion handled (*i.e.* least expected shortest path). In this operation, we stated that it is better to use the edge e if it allows reaching the platform to from ds in a better time than its current arrival time.

In the following, we explain the process of building a candidate solution cs based on another solution s and an edge e. As already mentioned, a solution has a travel time distribution that contains a list of discrete points where each discrete point has an expected arrival time and a probability to occur. An edge may have one or several weights depending whether it is time dependent or not. An edge's weight may take one value or a random value within a certain range depending whether the edge is considered as stochastic or not. Based on these requirements, building a candidate solution cs will consist of propagating the time distribution of s with respect to the travel time distribution of the edge e.

In the following we give an example of building such a candidate solution illustrated in Figure 6.5. We assume that s is a solution from the departure station ds to the platform p_1. s has a travel time distribution that consists of three discrete points. The first discrete point $(10, 0.3)$ contains an arrival time of 10 and a probability of occurrence at 0.3. In another term, we define that there is a probability of 0.3 to arrive at the platform p_1 at 10. This latter represents an arrival time (hh:mm:ss) that has been transformed into an integer value for illustration purposes.

The edge e considered in this example is a time dependent edge with two weights. The first weight corresponds to the departure time 10 and the second departure is at 30. 10 and 30 represent a departure time in the form (hh:mm:ss). The edge e is also a stochastic edge; thus, its travel time may take several values, each associated

with a probability.

Figure 6.5 – Stochastic example 1

solution s time distribution: $(\{10, 0.3\}\{15, 0.4\}\{20, 0.3\})$

edge e time distribution:

$(10 \rightarrow \{\{1, 0.3\}\{2, 0.4\}\{3, 0.3\}\}, 30 \rightarrow \{\{1, 0.3\}\{2, 0.4\}\{3, 0.3\}\})$

Building the time distribution of the candidate solution cs consists of selecting each discrete point in s and propagate it along e. When propagating a discrete point dp, the probability associated with dp will be affected since e is a stochastic edge, and the arrival time will also be affected by the travel time of e.

For each discrete point, the right weight w of e should be identified since e is a time dependent edge. In our example, the selected weight for e will be 10 as all the value for s (*i.e.* 10, 15, 20) are higher than 10 and lower than 30. After selecting the appropriate weight, dp should be mated with all the discrete points belonging to w. For each two pairs of discrete points (dps, dpe), a new discrete point dpn should be created for the candidate solution cs. The probability value in dpn is computed by multiplying the probability in dps by the probability in dpe. The arrival time in dpn will be the sum of the departure time in dps and the travel time in dpe.

Therefore, the new constructed solution will have a time distribution resulting from combining the arrival time distribution of s and the travel time distribution of e. The resulting arrival time distribution in this example will be the following: cs time distribution:

({11,0.09}{12,0.12}{13,0.09}{16,0.12}

{17,0.16}{18,0.12}{21,0.09}{22,0.12}{23,0.09})

As it is noticed, the sum of probabilities of all the discrete points inside one solution is always equal to one.

It is worth mentioning that edges in this work can either be correlated (*i.e.* random variables are stochastically dependent) or independent. In the former, as previously explained, the pointwise sum is used to merge two distributions. In the latter, the convolution product can be used instead. In real world situations, and especially in collective transport modes, the disruptions delaying a vehicle will also cause delays for the successors vehicles. Thus the links are more likely to be stochastically dependent.

After computing the travel time distribution of the candidate solution *cs*, a comparison has to be done between *cs* and the best solution *bs* ever registered in the platform *to*. The best solution *bs* is replaced by *cs* if the mean of the travel time distribution of *cs* is smaller than its corresponding in *bs*. Otherwise, *bs* remains unchanged. Computing the mean is done by summing the result of the multiplication between the arrival time and its associated probability over all the discrete points of a solution weighted by their respective probability. When a solution is replaced by a candidate solution *cs*, this latter is added to the priority queue in order to re-accomplish the relaxation phase on *cs*.

Edge could also be stochastic but time independent. More precisely, the edge in this case has only one weight and not a set of weights with respect to a set of departure times. The travel time on the edges takes in this case a random value within a predefined range. As in the previous case, nothing will change in the algorithm except that identifying the right weight will be omitted since there is one and only one weight.

Another case may happen when edges are time independent and static. For instance, one can say that the travel time on a transfer edge is 5 minutes regardless the time at which this edge is crossed and no matter the situation is. In this case, the algorithm could be easily adapted by considering that this edge has one and only one discrete point *dp*. The probability inside *dp* is set to 1 and the travel time is set to the predefined fixed value of the edge.

The algorithm terminates when the priority queue becomes empty or when a platform belonging to the arrival station has been reached. At this stage, we will have a solution representing the least expected shortest path to come from the departure station *ds* to the arrival station *as*. Furthermore, the solution will contain all the set of discrete points containing all the expected departure times and their probabilities. Such information can be useful for users who want to know the best

and worst arrival time of the proposed path with their corresponding probabilities.

Algorithm 8: Pseudo code of the proposed Monocriteria Stochastic Algorithm

```
input : Departure station ds
        Arrival station as
        Departure time dt
output: The least expected shortest path from ds to as
1  begin
       // Initialization
2      Initialize a priority queue pq
3      Initialize the solutions corresponding to the platforms
4      Initialize the solutions of the platforms belonging to ds
       // Relaxation
5      while pq is not empty do
6          Solution s = pq.poll()
7          Node n = s.getPlatform()
8          le = n.getAdjacentEdges()
9          foreach e ∈ le do
10             Node to = e.getTo()
11             Solution cs = s.constructCandidateSolution(e)
12             Boolean added = to.add(cs)
13             if added == true then
14                 if to.getParentStation().getId() == as.getId() then
15                     as.add(cs)
16                     return
17                 else
18                     pq.add(cs)
19                 end
20             else
21             end
22         end
23     end
24 end
```

C Example of Stochastic Routing

We show in Figure 6.6 an example consisting of three platforms A, B, and C. We assume that a passenger has reached the node A via a transfer from another transportation mode or from his/her departure place. The passenger's arrival time distribution at the node A is known. That is, the passenger can arrive at the node

A at any time between 7:00 and 7:10 according to a normal distribution function. We also assume that four vehicles will depart from the node A. The departure time of each vehicle is a random variable that varies according to a normal distribution law between 7:00 and 7:10, or 7:05 and 7:15, or 7:10 and 7:20, or 7:15 and 7:25. The travel time of each vehicle between nodes A and B is also a random variable between 1 and 3. We also assume that the travel time between the two nodes does not depend on the departure time of the vehicles. However, the departure time of the vehicles between nodes B and C depends on the departure time of the vehicles between A and B. The question we want to address in this example consists of computing the arrival time distribution of the passenger at the target node C.

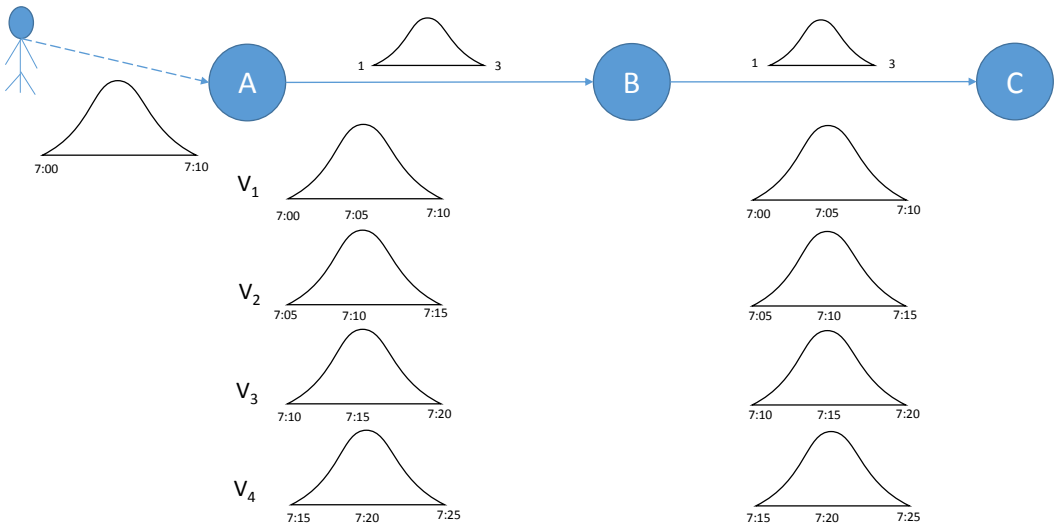


Figure 6.6 – Example of stochastic routing

As previously mentioned, the first step is to compute the arrival time distribution at the node B. To do so, a merge has to be done between the passenger's arrival time distribution at the node A, the departure time of vehicles at the node A, and the travel time distribution between the node A and B. The merge operation is done after discretizing each of the distributions. The result of this operation will be a discrete distribution with 7:00 as lower bound and 7:23 as upper bound.

After computing the arrival time distribution at the node B, propagating this distribution to the node C has to be performed. In case that random variables corresponding to departure times of vehicles are independent, the above mentioned merge operation can be performed. If random variables are dependent, merging probabilities turns out to be difficult. That is, if a vehicle departs from the node A earlier or later than its scheduled departure, its departure time distribution at the node B will change. For instance, if the vehicle V_1 departs from A at 7:10, its departure time at the node B cannot vary within [7:05; 7:15]; the successor departure distribution has rather to be updated.

Instead of merging the arrival time distribution at the node B with the corresponding departure time distributions, we introduce an additional random variable to account for the waiting time of a vehicle at the node B. The distribution function of this variable depends on the state of the vehicles at the node A, and the state of the node B. One can imagine a large table in which all possible distribution functions of the waiting time variables are stored. Such an information can be computed based on historical data and experts' observations.

Computing the arrival time distribution at the node C can then be done by merging the arrival time distribution at the node B, the intermediate dynamic waiting time distribution at B, and the travel time distribution of the edge (B, C).

D Experimental Results

To evaluate this work, a routing application based on the real data of the French Region Île-de-France has also been developed. The aim of the experimental phase is to study the performance of the proposed algorithm in solving the emerging routing problem. A comparison is made with the single criterion algorithm that has been introduced in chapter 4. This algorithm computes the path that minimises the deterministic travel time. This latter refers to the theoretical values that can be found in timetables information. The comparison is made to validate the importance of integrating stochasticity inside a route planning system. Both, deterministic and stochastic algorithms are compared with a deterministic algorithm applied a posteriori. This latter refers to the optimal solution since it is computed after knowing the exact state of the system.

The comparison is done by solving 1,000 routing queries, each having a departure and arrival stations and a departure time uniformly generated at random. For each query, 1,000 scenarios are considered. A scenario is an instance of the multimodal graph and is constructed based on real time traffic data. Instances are uniformly picked at random. Two indicators are used for comparisons: the average running time and the quality of solutions with respect to the deterministic algorithm applied a posteriori (*i.e.* after fixing a scenario and having the exact travel time values); in this case, the optimal path is obtained.

The first result obtained is that with the stochastic approach, the travel time distribution of the least expected shortest path is also provided. Thus, in contrast to the deterministic approach, passengers can have an idea about the robustness of the provided route. This information is very crucial, since some passengers may prefer to use other transport means in case that the travel time distribution reflects a high degradation of the solution.

Assessing the deterministic approach applied a priori shows that in most cases,

the quality of the solutions may become very bad in case of small variations. Over all scenarios tested, the results in Table 6.1 indicate that the path resulting from the deterministic approach applied a priori is not usually the same as the path resulting from the deterministic approach applied a posteriori. This means that, the travel time variations on edges may cause high degradation for the solutions provided by the deterministic approach applied a priori. When it comes to the stochastic algorithm, the results indicate that in most cases, the resulting routes are more robust. These experimentations have shown that integrating stochasticity will highly provide passengers with more robust itineraries than the deterministic approach.

Assessing the time performance of the proposed stochastic algorithm indicates also that the running time does not constitute a bottleneck for the stochastic algorithm to be integrated within a real world route planning system. Over all scenarios tested, the running time does not exceed one second.

Table 6.1 – Experimental results of the monocriteria stochastic algorithm on 1,000 routing queries with 1,000 stochastic scenarios each

Algorithm	**Time performance (ms)**			**GAP %**		
	◊	□	⋆	◊	□	⋆
- Deterministic	255	20	292	19	0	30
- Stochastic	754	280	986	6	0	11

GAP w.r.t. optimal solutions on a posteriori fixed scenarios;
◊: average; □ minimum; ⋆: maximum

E Real Case Scenario

We show in Figure 6.7 a real world routing request that consists of going from the station "LA DEFENSE" to the station "TOUR EIFFEL" at 09 : 15.

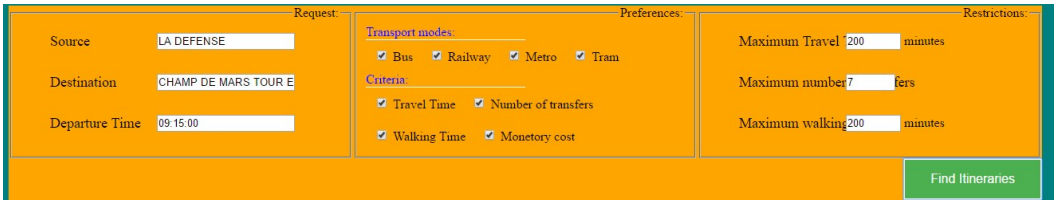


Figure 6.7 – Real world routing request in Ile-de-France Region

After applying the stochastic routing algorithm, the least expected shortest path is computed and its arrival time distribution is also provided (see Figure 6.8). The computing time to get it was 700 milliseconds.

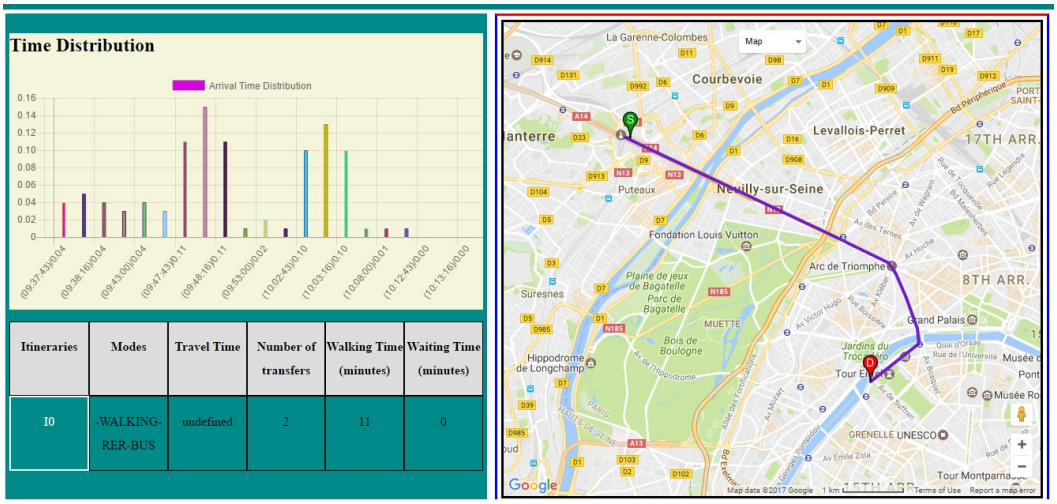


Figure 6.8 – The arrival time distribution for the least expected shorted path on the real world routing request

For more details about the path found, see Figure 6.9.

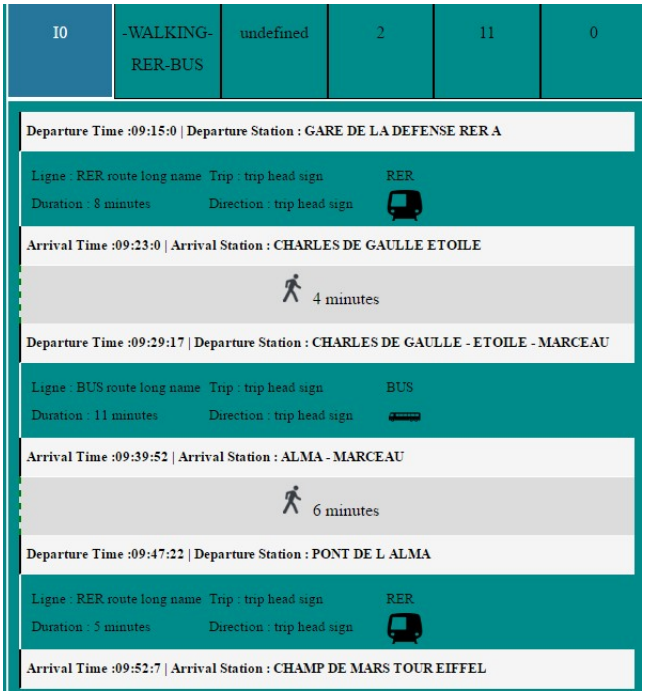


Figure 6.9 – Path details of the solution on the real world routing request

In the stochastic algorithm, the only criterion taken into consideration here is the expected travel time. Since the developed algorithm is quite generic, the lower

bound and upper bound have been also solely minimised.

Using the lower bound of the travel distributions as a metric to compare the solutions is not robust. This can be explained by the fact that in most real world situations, the vehicles do not usually arrive earlier at the platforms.

Using the upper bound of the travel time distributions provides more robust solutions. However, that robustness is usually at the expense of having additional travel time, even in the case without traffic disruptions.

The standard deviation of the travel time distributions has also been used as a metric to compare the solutions. Results show that in this case, the most robust path is obtained. However, its travel time may exceed one day. It is clear, that taking this path to ensure robustness is not a practical solution.

We show, in Figure 6.10 and Figure 6.11 the path that minimises the standard deviation and its travel time distribution.

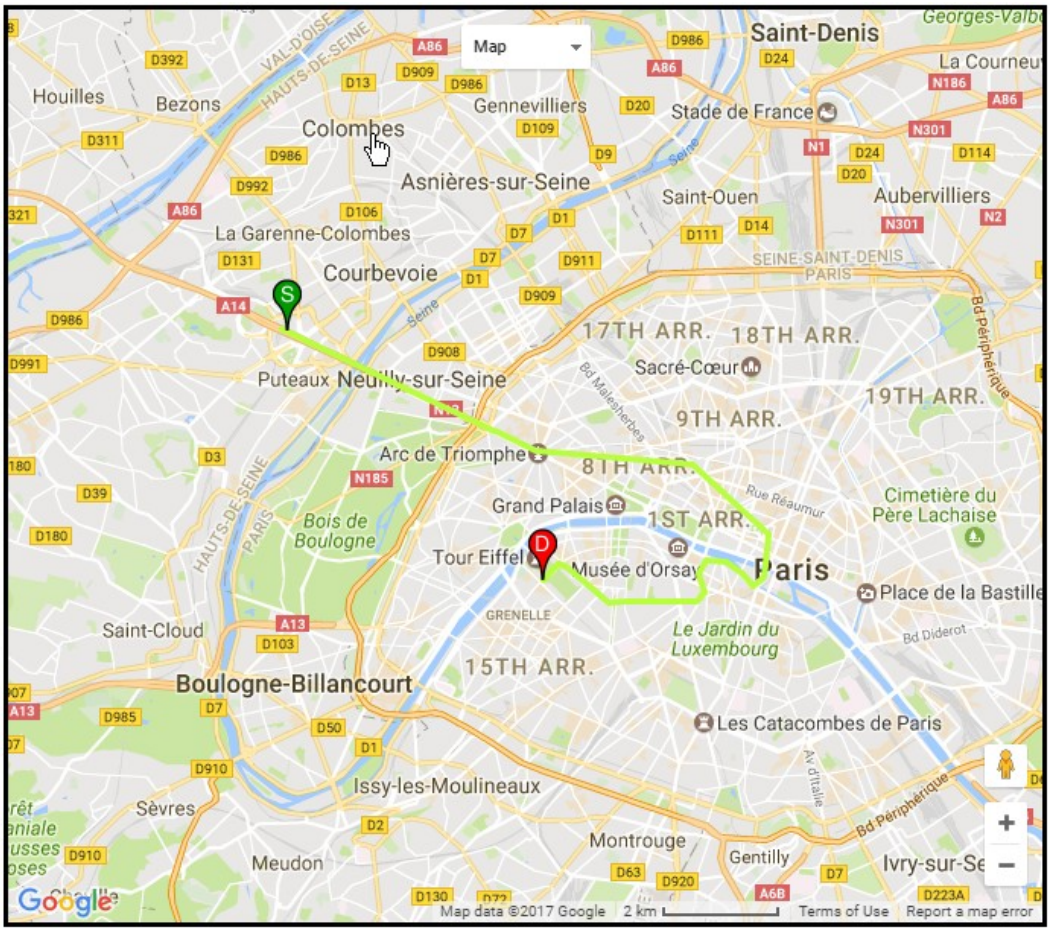


Figure 6.10 – The path that minimizes the standard deviation

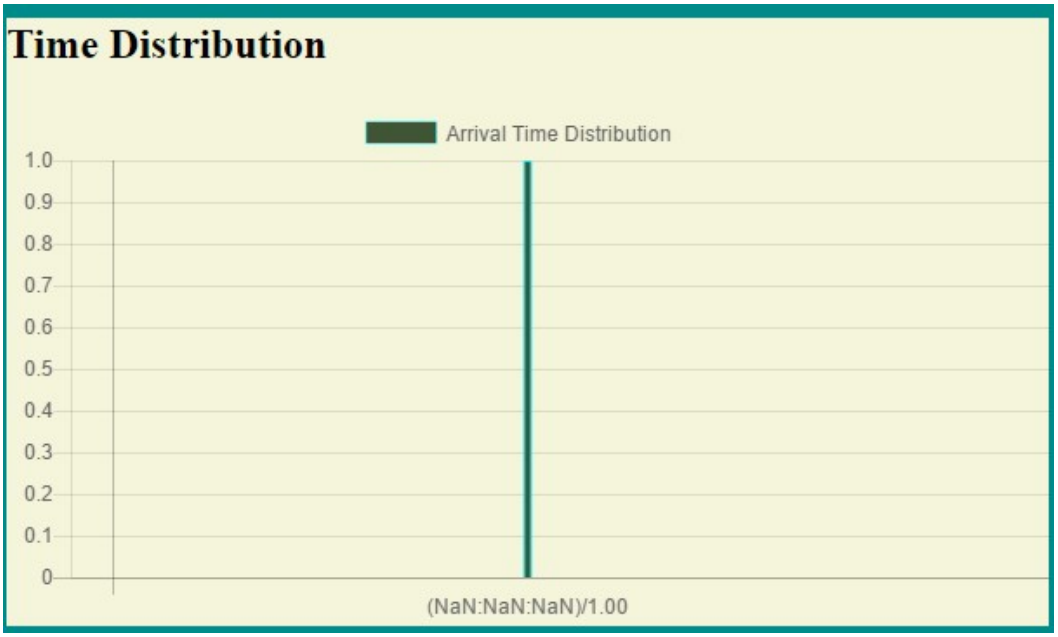


Figure 6.11 – The arrival time distribution for the path that minimises the standard deviation

6.6 Multicriteria Stochastic Algorithm

As mentioned in previous sections, several criteria can be optimized. The developed stochastic algorithm is, however, adapted for only optimizing one criterion which is the travel time. Integrating the lower and upper bounds, as well as the standard deviation in a multicriteria route planner can provide passengers with many transportation alternatives to go from one place to another. The passenger will then be responsible for choosing his most preferred one.

A Problem to Consider

The studied routing problem in this chapter consists therefore of solving the multicriteria shortest path problem in a time dependent stochastic multimodal network. Criteria that are considered in this chapter are: the expected arrival time β, the best (minimum) arrival time α, the worst (maximum) arrival time γ, and the standard deviation of the arrival time of the path δ.

The formula related to the computation of each of these objectives are respec-

tively presented as follows:

$$Z(path) = \beta(path) = \varpi(D) \tag{6.2}$$

where D represents the node corresponding to the arrival station, the objective β to minimize represents the average (expected) arrival time of the journey, and ϖ is the function responsible for computing the average arrival time at a given node. β is similar to f_1 in Chapter 3 but for the stochastic context.

$$Z(path) = \alpha(path) = \rho(D) \tag{6.3}$$

where α refers to minimizing the best (minimum) arrival time of the journey and ρ is the function responsible for computing the minimum arrival time at a given node.

$$Z(path) = \gamma(path) = \varrho(D) \tag{6.4}$$

where γ refers to minimizing the worst (maximum) arrival time at the destination place and ϱ is the function responsible for computing the worst arrival time at a given node.

$$Z(path) = \delta(path) = \varsigma(D) \tag{6.5}$$

where δ refers to minimizing the standard deviation of the journey and ς is the function responsible for computing the standard deviation of the arrival time at given node.

The functions ϖ,ρ, ϱ, ς take a distribution function as an input. Their computation is also straightforward since ϖ returns the average of the distribution function, ρ returns the lower bound, ϱ returns the upper bound, and ς returns the standard deviation.

We introduce in the following a multicriteria algorithm that can compute multicriteria shortest paths with respect to several criteria in a time variant stochastic multimodal network.

Example: Let us imagine that s^* is the least expected shortest path to go from a station S_1 to S_2. Let us also imagine that the arrival time distribution of s^* consists of three discrete points ($\{0.3, 12:30\}\{0.4, 14:00\}\{0.3, 15:30\}$). As can be noticed, the mean arrival time of $s*$ is $14:00$. Let us now imagine that s' is another

solution to go from S_1 to S_2. We assume also that the distribution function of $s\prime$ consists of the following discrete points $(\{0.3, 13:30\}\{0.4, 14:30\}\{0.3, 15:00\})$.

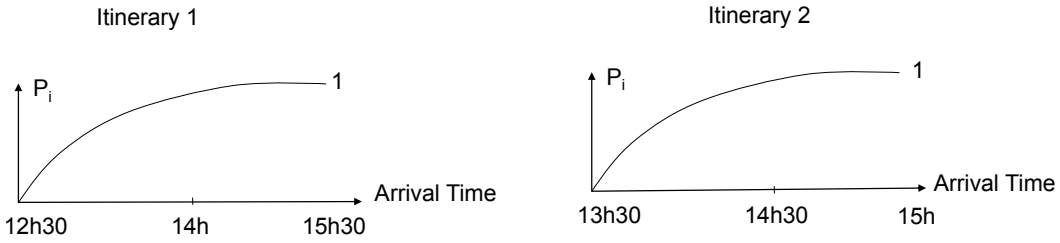


Figure 6.12 – Importance of multicriteria analysis

By minimizing the expected arrival time, it is clear that s^* dominates s'. Therefore, s' will be omitted during the monocriteria algorithm's search process. However, by looking carefully, it can be noticed that in the worst case, s' is better than s^*. In many cases, the passengers may also be very interested in the value of the arrival time in the worst case or even in the best case, and then take in consideration a global strategy to decide its travel.

We believe therefore that providing passengers with such nondominated solutions will largely enhance the ultimate mobility service. Integrating such additional criteria into the routing process requires some adaptations at the previously discussed routing algorithm.

Criteria Analysis: In multicriteria optimization, it is important to study the relation between criteria in order to ensure the absence of correlation between them. Correlated criteria can be usually reduced to one criterion. Indeed, reducing the number of considered criteria does often reduce the running time required to solve the problem.

To analyse the criteria, the monocriteria algorithm proposed in previous section is applied to solve several routing request. The value of each of the following criteria, the mean arrival, the earliest arrival, the latest arrival and the standard deviation of the time of arrival is then stored.

As can be seen from Figure 6.13, the criteria considered are completely uncorrelated. This validates that all such criteria have to be integrated within a global multicriteria routing analysis.

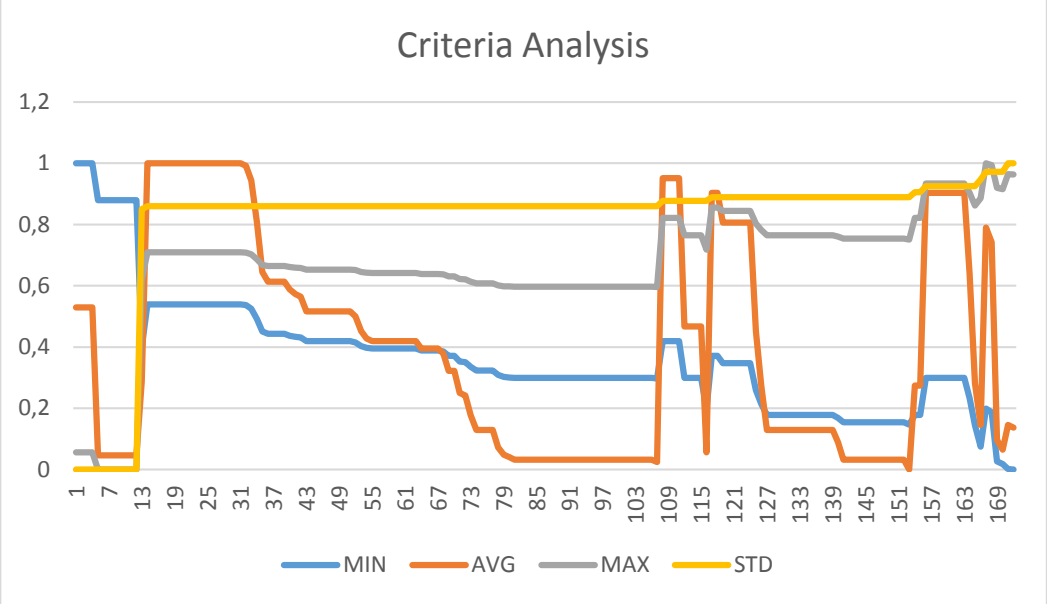


Figure 6.13 – Analysing criteria in stochastic environment

B Multicriteria Routing Algorithm

The first important modification while developing a multicriteria routing algorithm consists of replacing the single solution stored in a platform p with a set of nondominated solutions representing nondominated paths from the departure station ds to p.

The second adaptation lies in the comparison operation between two solutions. Several criteria instead of one criterion have to be compared in order to decide whether the candidate solution will enter the list of nondominated solutions or not.

In contrast to the single criterion algorithm, in a multicriteria context, the algorithm cannot terminate when a platform belonging to the arrival station is reached. Indeed, more than one nondominated solution may be found between two stations. Therefore, the one and only one stopping criterion is when the priority queue becomes empty.

We present in Algorithm 9 the pseudo code of the routing algorithm that has been developed for solving multiple criteria shortest paths in time dependent and

stochastic multimodal networks.

Algorithm 9: Pseudo code of the Multicriteria Routing Algorithm in a time dependent stochastic network

input : Departure station ds, Arrival station as, Departure time dt
A set of criteria: $C = \{$mean, earliest arrival, latest arrival, standard deviation$\}$
output: Nondominated paths from ds to as with respect to dt and C

```
1  begin
       // Initialization
2      Initialize a priority queue pq
3      Initialize the list of nondominated solutions of ds
4      Initialize the list of nondominated solutions of all other nodes in
       G(V,E)
       // Relaxation
5      while pq is not empty do
6          Solution s = pq.poll()
7          Node n = s.getPlatforms()
8          l_e{} = n.getAdjacentEdges()
9          foreach e ∈ l_e do
10             Node to = e.getTo()
11             Solution cs = s.constructCandidateSolution(e)
12             Boolean added = to.add(cs);
13             if added == true then
14                 if to.getParentStation().getId() == as.getId() then
15                     as.add(cs)
16                 else
17                     pq.add(cs)
18                 end
19             end
20         end
21     end
       // Visualisation
22     foreach Itinerary i ∈ as.getNonDominatedSolutions() do
23         Path p = i.reconstructPath()
24         Visualize(p);
25     end
26 end
```

As can be noticed, the algorithm has three main phases: the initialization, the relaxation and the visualization.

At first, a priority queue pq is initialized. Elements in pq represent the solutions

to go from the departure station ds to a platform p. A solution is an entity having several properties such as the arrival time distribution, the number of transfers and the walking time. A solution s is higher in priority than another solution $s\prime$ if the expected value of the arrival time distribution of s is lower.

Each platform p in the graph is initialized with a list of solutions representing the set of nondominated paths to go from ds to p. At the beginning, the arrival time distributions of all the initial solutions contain one single discrete point dp. The value of all criteria inside the initial solutions is set to ∞.

The platforms belonging to the departure station are initialized differently with respect to the departure time. In the relaxation phase, the algorithm takes the first element in the queue and relaxes its adjacent platforms. These latter are chosen so that the time dependency of the edge is also respected.

The propagation of the probabilities is done by merging the arrival time distributions of the departure platform with the distribution of the departure time of the vehicles and the distribution of the travel times along edges. The merge can differ according to whether the distributions are dependent or not.

After merging the probabilities, the list of nondominated solutions of each adjacent platform is also updated. Solutions are removed if they are dominated by the newly generated solutions. In this case, it can be said that a better solution can be found between the departure station and the adjacent platform.

The algorithm terminates when the priority queue becomes empty. This ensures that no other nondominated solutions can be found between the departure station and the arrival station.

C Experimental Results

A web-based routing application has been developed to assess the performance of the proposed algorithm. Real world data of the French Region Ile-de-France have been used during implementations.

A comparison has been made between the proposed multicriteria algorithm and a single criterion deterministic algorithm that does not include stochasticity inside its search process, and the single criterion routing algorithm proposed in the previous section.

Experimentations were done by solving 1,000 routing requests. The departure and arrival stations and the departure time are uniformly picked at random.

Results in Table 6.2 show that the average running time of the proposed multi-

criteria algorithm to optimize three criteria (mean, earliest and latest) is 25 seconds in average case and may increase to 30 seconds in the worst case.

In case that only two criteria are considered, the average running time may decrease to 15 seconds in average case to 25 seconds in the worst case.

Results have also indicated that the average number of nondominated paths is seven when two criteria are considered and 10 when all criteria are optimized.

To assess the robustness of the proposed algorithm, several scenarios are considered. Each scenario corresponds to a real case state of the multimodal graph where the travel time on edges may take random values. A graph state is constructed based on real time traffic data.

1,000 scenarios are considered, and for each scenario 1,000 routing requests is solved. Results obtained validates the fact that paths obtained from the multicriteria approach are more robust in comparison with both the single criterion approach that integrates stochasticity and the deterministic algorithm that disregard stochasticity. Besides, the multicriteria approach usually provide passengers with several alternatives to go from the departure to the destination station.

Table 6.2 – Experimental results of the multicriteria stochastic algorithm on 1,000 routing queries with 1,000 stochastic scenarios each

Criteria	**Running time in seconds**			**Number of solutions**		
	$>$	$<$	˜	$>$	$<$	˜
(α, β)	3	25	15	1	21	7
(α, γ)	4	21	14	1	19	8
(α, β, γ)	11	30	25	1	25	10

α: earliest arrival time β: mean arrival time γ: latest arrival time
Number of solutions: $>$ Minimal $<$ Maximal ˜ Average

D Real Case Scenario

We present in this section a real case scenario that consists of going from the station "LA DEFENSE" to the station "TOUR EIFFEL". Several criteria are optimized and the value of each criterion is presented. Non dominated paths are also visualized.

In Table 6.3, the expected, the earliest and the latest arrival time are optimized as three criteria all together. Three nondominated paths are found mentioned as I_1, I_2 and I_3. For each solution, we compute the value of all criteria.

Table 6.3 – Minimizing the minimum, the average and the maximum arrival time

Itinerary	α	β	γ	δ	ζ	η
I_1	09:37:43	09:52:7	10:13:16	9	2	11
I_2	09:40:43	09:53:36	10:11:16	5	1	4
I_3	09:52:43	10:01:10	10:08:16	4	2	7

α: minimum arrival time β: average arrival time γ: maximum arrival time
δ: standard deviation ζ: number of transfers η: walking time in minutes

Based on these results, the three nondominated solution are visualized. Table 6.3 shows evaluation on 6 criteria but the optimization is only done on 3 criteria.

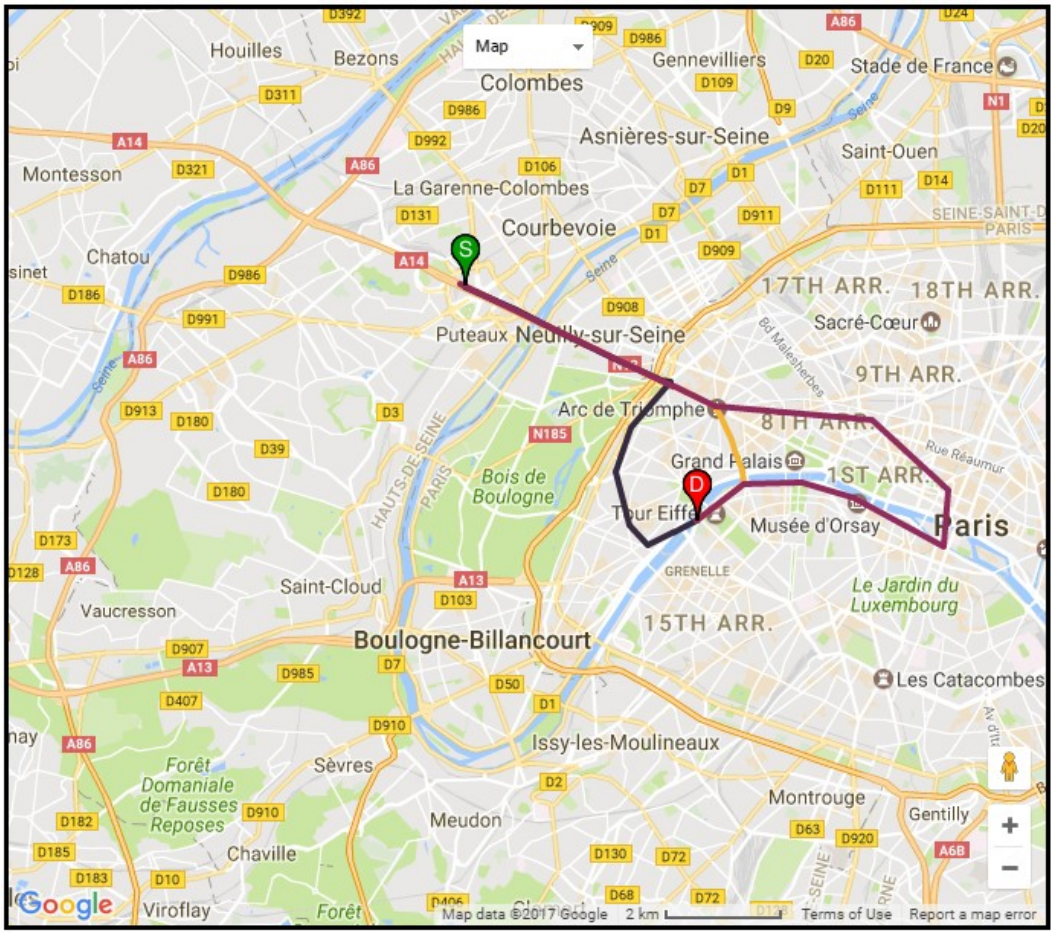


Figure 6.14 – The set of three nondominated paths w.r.t. the minimum, the average and the maximum travel time

The average arrival time and the standard deviation of the arrival time have been also optimized. Seven nondominated solutions have been found. Results can

be seen in Table 6.4. The value of 6 criteria are reported in the table, but the optimization is only made on 2.

Table 6.4 – Minimizing only the average arrival time and the standard deviation of the arrival time

Itinerary	α	β	γ	δ	ζ	η
I_1	09:49:41	09:56:34	10:11:40	4	2	19
I_2	09:37:43	09:52:7	10:13:16	9	2	11
I_3	09:40:43	09:53:36	10:11:16	5	1	4
I_4	09:52:43	10:01:10	10:08:16	4	2	7
I_5	10:02:43	10:04:30	10:08:16	2	2	9
I_6	10:17:43	10:18:0	10:18:16	0	2	9
I_7	00:00:0	00:00:0	596523:14:7	0	2	7

α: minimum arrival time β: average arrival time γ: maximum arrival time
δ: standard deviation ζ: number of transfers η: walking time in minutes

The seven paths are also visualized in Figure 6.15.

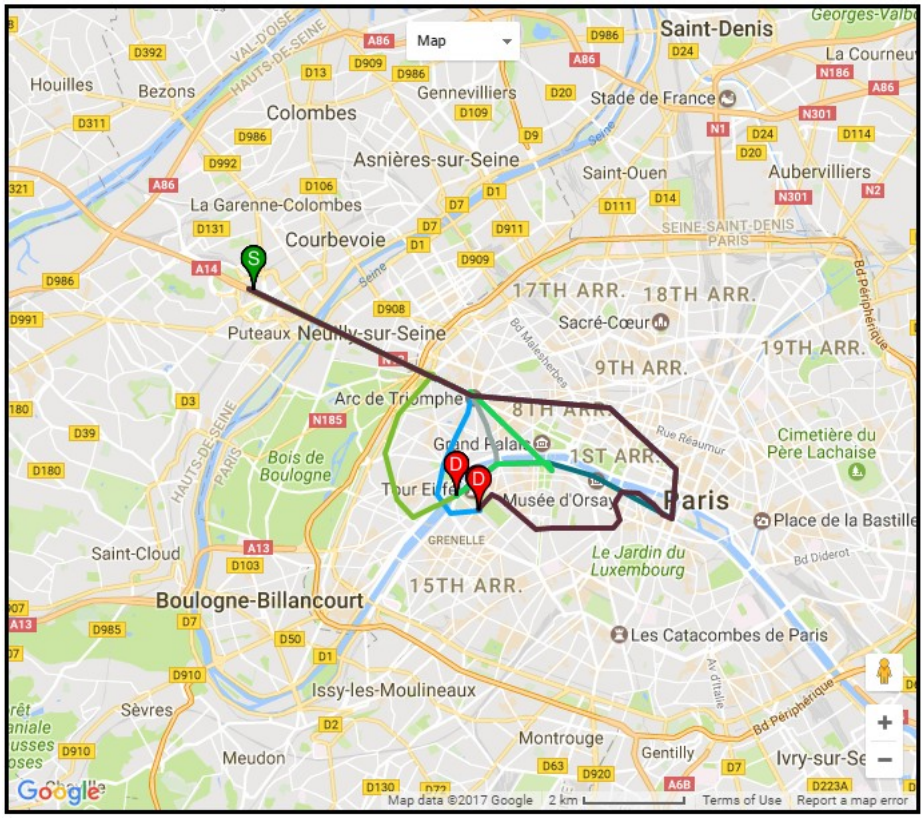


Figure 6.15 – The set of the seven nondominated paths w.r.t. the average arrival time and the standard deviation

The minimum, the average and the standard deviation of the arrival time are also optimized. Seven solutions have been found (see Table 6.5). The value of 6 criteria are reported in the table, but only 3 criteria are optimized.

Table 6.5 – Minimizing the minimum, the average of the arrival time and the standard deviation of the arrival time

Itinerary	α	β	γ	δ	ζ	η
I_1	09:49:41	09:56:34	10:11:40	4	2	19
I_2	09:37:43	09:52:7	10:13:16	9	2	11
I_3	09:40:43	09:53:36	10:11:16	5	1	4
I_4	09:52:43	10:01:10	10:08:16	4	2	7
I_5	10:02:43	10:04:30	10:08:16	2	2	9
I_6	10:17:43	10:18:0	10:18:16	0	2	9
I_7	00:00:0	596523:14:7	00:00:0	0	2	7

α: minimum arrival time β: average arrival time γ: maximum arrival time
δ: standard deviation ζ: number of transfers η: walking time in minutes

6.7 Synthesis

We studied in this chapter several routing issues in time dependent and stochastic multimodal networks. A single criterion algorithm was first developed. The expected travel time was used as a metric to compare solutions. Several points have to be considered in stochastic networks such as defining the uncertain components, describing their variation, propagating the probabilities while the algorithm accomplished its search process, then evaluating and comparing the solutions. We have indicated that propagating the probabilities does depend on whether the random variables are completely independent or dependent such as in railway network. Experimental results showed the high performance of the proposed algorithm in providing more robust itineraries in comparison with a classical deterministic algorithm.

Besides, a multiple criteria algorithm was introduced. Criteria considered are the expected arrival time, the earliest arrival, the latest departure and the standard deviation of the arrival time. In contrast to the single criterion algorithm, each node contains a set of nondominated solutions, which are updated during the search phase. Experimental results indicate that the algorithm succeeded at providing several nondominated solutions, which are more robust in comparison with results obtained from the other algorithms. Moreover, several solutions are always obtained. This can be very useful from a transport operator point of view in order to efficiently reroute passengers in case of disruptions.

Chapter 7

Conclusion and Perspectives

This thesis is the result of three years of theoretical and practical work at the Technological Research Institute IRT SystemX. A collaboration has been made with the University of Technology of Belfort-Montbéliard (UTBM), the University of Bourgogne Franche Comte (UBFC), and Alstom transport as an industrial partner. This work will be used as a part of the smart supervision module that has been developed by Alstom. Below, the main contributions of this thesis are summarised, and the future works are outlined.

7.1 Summary of Contributions

In this thesis, we have focused on solving route planning issues in Multimodal Transportation Networks (MTN). This latter refers to a combination between individual transportation networks that offer continuous services (*e.g.* pedestrian, bike, road) and collective transport modes that work according to predefined timetables (*e.g.* bus, tram, train).

We began this manuscript by highlighting the main motivations behind building an Advanced Travellers Information System whereby passengers' routing queries are efficiently answered in ideal cases, as well as, when the transportation system encounters disruptions.

Besides, we indicated that it is important for any route planning system to integrate the various needs and preferences of passengers. Moreover, the dynamic aspect of the transport system should be considered. Indeed, the transport system may change due to anticipated events such as closure of roads and strikes or unanticipated events such as technical problems, accidents, traffic jams, bad weather conditions or terrorist attacks.

Additionally, we showed that when providing passengers with routes, it is important to take into consideration the various capacities of the vehicles, which may also dynamically change as a function of time.

Furthermore, we indicated that answering a passenger's request should be done within a reasonable amount of computational time. Therefore, the computational

complexity of the routing algorithms should not be disregarded.

It has been indicated in Chapter 2 that several route planning systems are nowadays used by a large number of passengers or transport operators. However, such systems may not always be capable of solving routing issues under some circumstances either due to limitations of the used modelling approach or the low efficiency of routing algorithms themselves.

To build an intelligent route planning system, we proposed in Chapter 3 a novel modelling approach for representing the multimodal transport networks. Both individual and collective transport modes are considered. The representation allows considering the multimodality aspect of the transport system. It also allows handling the dynamic and stochastic nature of the various transportation modes. Moreover, the access points, the platforms, the routes and the vehicles are represented in the model via separated entities. As a result, more advanced situations such as rerouting in case of disruption can be dealt with. Besides, a general mathematical formulation is introduced based on the proposed modelling approach in order to formally describe various routing issues.

Based on the proposed modelling approach, we propose in Chapter 4 an advanced routing algorithm for solving multicriteria routing problems faced by travellers while going from one place to another. Several accelerating strategies are also proposed in the goal of decreasing the algorithm's search space. Although speed up techniques succeed at enhancing the algorithm's computational time, the real world situations require fast answers. Therefore, we propose in Chapter 5 to solve routing issues using metaheuristics. By doing so, high quality solutions can be obtained within a reasonable amount of time. Besides, we overcome the drawbacks of standard exact algorithms regarding their flexibilities in handling stochasticity.

The proposed metaheuristic to solve the multicriteria routing issue is a combination between a Genetic Algorithm and Variable Neighborhood Search. The resulting Memetic Algorithm takes the advantageous of Genetic Algorithm in efficiently exploiting a large search space and also benefits from the capacity of Variable Neighbourhood Search in intelligently exploring promising areas within the search space.

To assess the performance of both exact and heuristic approaches, a web-based routing system has been developed and real world routing queries have been solved. Data refer to the French Region Ile-de-France that includes the city of Paris and its suburbs. General Transit Feed Specification (GTFS) data are used to build collective transit networks and Open Street Map (OSM) has been used to extract data related to individual transportation modes. Results have indicated that the proposed routing system is efficient enough to answer real world routing queries.

In Chapter 6, we first show that a small delay of one second in the departure

time of a vehicle may cause a huge variation in the travel time metric of solutions. Therefore, it is primal to provide passengers with information about the distribution of the arrival time at the target place. To do so, random variables should be assigned to edges; this does also depend on time since several travel time distributions can be assigned to an edge w.r.t. the time the edge is crossed. Secondly, a stochastic routing algorithm should be designed whereby propagating probabilities in the transport network remains correct. For this purpose, we propose a new algorithm to compute the path(s) that minimises the expected shortest path. A special attention is given to the propagation of probabilities since random variables are dependent. A delay occurring in a metro may cause delays to other transport modes. Besides, we extend the algorithm to work in a multicriteria context. The best, the average, the worst and the standard deviation of the arrival time of a path are the criteria taken into consideration. The algorithm has been implemented and tested on real world situations. The experimental phase shows that more robust solutions can be obtained by performing a multicriteria analysis within a stochastic environment. The results are also integrated in the final multimodal route planning system.

Finally, the works described in those various chapters were published as papers in two international journals with impact factor, seven international conferences and one national conference. They were also presented in the form of three posters and four videos (see the list of achievements in Chapter 7.2).

7.2 Future Works

Several works have been planned to be done in the future. In the first place, improvements of the proposed approaches are to be made.

Although the proposed modelling approach succeeded at handling basic and complex routing issues, extensions may be added in order to account for novel transportation modes (*i.e.* autonomous vehicle and sky car). Other novel mobility services may also be taken into consideration such as sharing of resources between several users (*e.g.* connecting people who possess underutilized resources with those wanting to rent vehicles for short periods of time). These issues require integrating users and other additional information during the modelling phase.

The performance of the proposed routing approaches may also be an area of improvement. For instance, the various parameters of metaheuristics can be studied more deeply by accomplishing additional statistical analysis and thus achieving their best tuning values. This issue of tuning also remains an important issue for the proposed algorithms to be easily and efficiently used for solving other optimization problems. This problem can be solved either empirically or theoretically. In the former, an additional tuning module may be built in order to automatically extract

the best parameters after performing an adequate number of executions. In the latter, an advanced optimization problem may be designed to optimally extract the values of the various parameters.

The computational complexity may also be enhanced by using parallel computing. The proposed memetic algorithm can be easily parallelized by making individuals separately evolve and intelligently exchange information about the search space to find better solutions. By doing so, the running time of the algorithm may be enhanced, and also the probability of finding better solutions will increase. For more information about the parallelisation of genetic algorithms, the reader can refer to [**?**] and [**?**].

Using Genetic Algorithm and Variable Neighbourhoud Search in particular is also a question to be tackled. Indeed, we have used in this work a Genetic Algorithm as a population based metaheuristic in order to solve various routing issues. Other population based methods such as Ant Colony Optimization [**?**] and Particle Swarm Optimization [**?**] can also be tested and compared with the proposed Genetic Algorithm. When it comes to single solution metaheuristics, the methods such as the Tabu Search [**?**], the Greedy Randomized Adaptive Search Procedure (GRASP) [**?**], and the Simulated Annealing [**?**] can be applied and compared with the proposed Variable Neighbourhood Search.

Besides, additional criteria may be considered while solving multiple criteria such as the CO2 emission. Analysing the performance of the proposed approaches while considering more than five criteria is also an area of investigation. Nowadays, proposing efficient multiobjective algorithms for solving optimization problems with respect to more than four objectives is a very important topic in the multiobjective community [**?**], [**?**]. Many researchers have started looking for efficient approaches whereby a large number of objectives is considered inside one optimization problem. Therefore, it is important to consider the efficiency of our algorithms in such context.

When it comes to handling stochasticity, a prediction module is to be integrated within the route planning system to provide routing algorithm with real travel time distributions along edges. The prediction module is the main subject of another PhD thesis [**?**] in the same project. Integrating both predictions and routing algorithms is planned to be done soon.

Finally, the proposed web-based routing application requires additional enhancements in order to run it in production for real world usage. For instance, building interfaces in order to populate our algorithms with real time data have to be done. In addition, building scalable visualization module has to be done in order to provide passengers with clear and relevant navigating information.

Implementation Details

Class Solution

```
import java.math.BigDecimal;
import java.util.Map.Entry;
import org.json.JSONException;
import org.json.JSONObject;
import fr.multimodalrouting.entity.gtfs.Route;
import fr.multimodalrouting.entity.gtfs.Trip;
import fr.multimodalrouting.utils.Util;

public class Solution {

   private Node parent;
   private Solution from;
   private Edge fromEdge;
   private int nbOfTransfers;
   private int walkingTime;
   private int waitingTime;
   private Trip trip;
   // For stochsticity
   private TimeDistribution arrivalTimeDist;
   private double averageArrivalTime;// mean
   private double standardDeviation;// ecartype
   private DiscretePoint minDP;// The discrete point that has the smallest
                       // arrival time
   private DiscretePoint maxDP;// The discrete point that has the largest
                       // arrival time
```

Class Node

```
public class Node {

   protected int id;
   protected List<Edge> adjacencies;
   protected List<Edge> incidents;
   protected Coordinate location;
   protected List<Solution> listNds;//List of nondominated solutions
   protected String name;
```

Class Weight

```
public class Weight {

   private int theoreticalTravelTime;
   private int numberOfTransfers;
   private int walkingTime;
   private Trip trip;
   private int currentLoad;
   private int maximumLoad;
   //Used to compute a distribution function for the edge
   private TimeDistribution timeDist;
```

Class Platform

```
package fr.multimodalrouting.entity.network;

import fr.multimodalrouting.entity.gtfs.Trip;
import fr.multimodalrouting.utils.Util;

public class Platform extends Node {

   private String name;
   private String desc;
   private Station parentStation;
   private int maximumCapacity;
   private int currentCapacity;
```

Building Departure Events

```
public void buildEvents() {
      String sql = "SELECT * from events";

      Connection conn = null;
      PreparedStatement pstmt = null;
      ResultSet rs = null;
      Platform departureStop = null;
      Platform arrivalStop = null;
      int departureTime = -1;
      int arrivalTime = -1;
      Trip trip = null;
      try {
```

```
        conn = DbUtil.getConn();
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();
        while (rs.next()) {
           departureStop = platforms.get(rs.getInt("departure_stop"));
           arrivalStop = platforms.get(rs.getInt("arrival_stop"));
           departureTime = rs.getInt("departure_time");
           arrivalTime = rs.getInt("arrival_time");
           trip = trips.get(rs.getInt("trip_id"));
           departureStop.linkByVehicle(arrivalStop, departureTime,
               arrivalTime, trip);
        }
     } catch (SQLException e) {
        e.printStackTrace();
     } finally {
        DbUtil.closed(rs, pstmt, conn);
     }
  }
```

Building and Linking Networks

```
package fr.multimodalrouting.main;

import java.io.IOException;

import fr.multimodalrouting.config.Config;
import
    fr.multimodalrouting.entity.scheduelbasedmodels.ScheduelBasedNetwork;
import fr.multimodalrouting.entity.unrestrictedmodes.BikeNetwork;
import fr.multimodalrouting.entity.unrestrictedmodes.PedestrianNetwork;
import fr.multimodalrouting.utils.DbUtil;

public class Main {

   public static ScheduelBasedNetwork sbn;
   public static PedestrianNetwork pn;
   public static BikeNetwork bn;

   public static void init(){
      sbn = new ScheduelBasedNetwork();
      pn = new PedestrianNetwork();
      bn = new BikeNetwork();
      DbUtil.loadConfig();
      Config.loadConfig();
   }
   public static void buildMultimodalNetwork() throws IOException{
```

```
    buildNetworks();
    linkNetworks();
}
public static void buildNetworks() throws IOException {
    buildSchedualBasedNetwork();
    buildPedestrianNetwork();
    //buildBikeNetwork();
    //setNearestPedestrianJunctions
    //sbn.setNearestPedestrianJunctions(pn.getJunctions());
    //Link Networks
    //sbn.linkToPedestrianNetwork(pn.getJunctions());
}

public static void linkNetworks(){
    linkSchedualBasedToPedestrianNetwork();
    //linkBikeRentalStationToNearestBikeJunction();
    //linkBikeRentalStationToNearestPedestrianJunction();
    System.out.println( " Start Checking if the pedestrian Network is
        disconnected" );
    System.out.println(pn.isDisconnected());

}
```

Data Base Model for Private and Public Networks

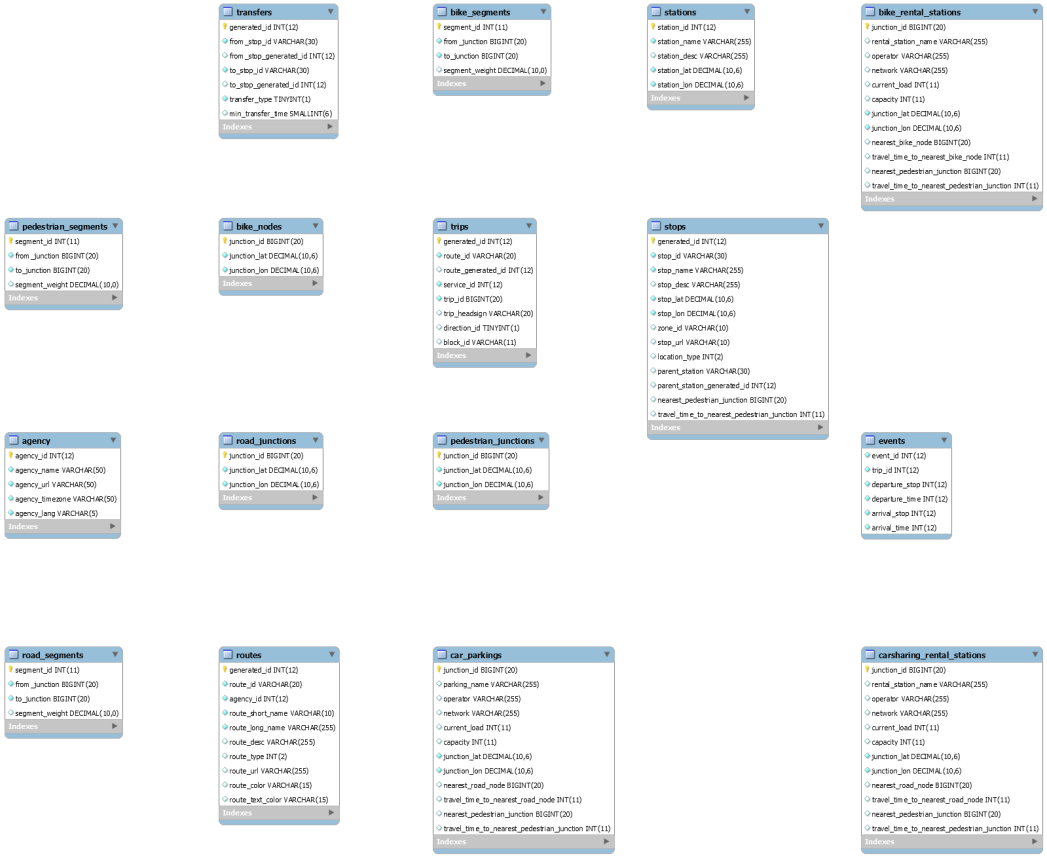


Figure 7.1 – DataBase Model for Private and Public Transport Modes

Achievements

International Journals

1. Omar Dib, Laurent Moalic, Marie-Ange Manier, and Alexandre Caminada. An advanced GA–VNS combination for multicriteria route planning in public transit networks. *Expert Systems with Applications*, Pergamon, 72:67–82, 2017 (Impact Factor: 3.928, SJR:Q1, CORE:B)

2. Omar Dib, Marie-Ange Manier, Laurent Moalic, and Alexandre Caminada. Combining vns with genetic algorithm to solve the one-to-one routing issue in road networks. *Computers & Operations Research*, Elsevier, 2015, (Impact Factor: 2.6, SJR:Q1)

International Conferences

1. Omar Dib, Alexandre Caminada, Marie-Ange Manier, and Laurent Moalic. A memetic algorithm for computing multicriteria shortest paths in stochastic multimodal networks. In *The Genetic and Evolutionary Computation Conference GECCO, July 15-19, Berlin*. GECCO, 2017, (Qualis:A1, ERA:A)

2. Omar Dib, Marie-Ange Manier, Laurent Moalic, and Alexandre Caminada. A multimodal transport network model and efficient algorithms for building advanced traveler information systems. *Transportation Research Procedia*, Elsevier, 22:134–143, 2017

3. Omar Dib, Alexandre Caminada, and Marie-Ange Manier. A genetic algorithm for solving the multicriteria routing problem in public transit networks. In *6th international conference on metaheuristics and nature inspired computing META , October 27-31 marrakech*. META16, 2016

4. Omar Dib, Marie-Ange Manier, and Laurent Moalic. Advanced modeling approach for computing multicriteria shortest paths in multimodal transportation networks. In *Intelligent Transportation Engineering (ICITE), IEEE International Conference on Intelligent Trasnportation Engineering*, pages 40–44. IEEE, 2016

5. Omar Dib, Marie-Ange Manier, and Alexandre Caminada. Memetic algorithm for computing shortest paths in multimodal transportation networks. *Transportation Research Procedia*, Elsevier,10:745–755, 2015

6. Omar Dib, Marie-Ange Manier, Laurent Moalic, and Alexandre Caminada. A hybrid metaheuristic for routing in road networks. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on Intelligent Trasnportation Systems*, pages 765–770. IEEE, 2015, (Qualis: B1)

7. Omar Dib, Alexandre Caminada, and Marie-Ange Manier. Genetic algorithm combined with variable neighborhood search to solve the one-to-one shortest path problem. In *11th Metaheuristics International Conference MIC, June 7-10 Agadir*. MIC15, 2015, (Qualis:B3)

National Conferences

1. Omar Dib, Alexandre Caminada, and Marie-Ange Manier. Memetic algorithm for solving one-to-one shortest path problem. In *16ème conférence ROADEF Société Française de Recherche Opérationnelle et Aide à la Décision, February 25-27 Marseille*. Roadeff, 2015

Presentations

1. Thesis Midterm Assessment
   (a) Date : UTBM - Belfort - March 17 2016
   (b) URL : Download

Posters

1. Dynamic rerouting in multimodal networks
   (a) Event : STIC Forum
   (b) Date : ENSTA - Paris Saclay - December 2 2014
   (c) URL : Download
2. Multicriteria Route Planning
   (a) Event : Thesis Day @SystemX
   (b) Date : IRT SystemX - February 2 2017
   (c) URL : Download
3. Stochastic Route Planning
   (a) Event : GECCO Conference
   (b) Date : Berlin - July 18 2017
   (c) URL : Download

Demonstrations

1. Dynamic rerouting in multimodal networks
   (a) Event : Thesis Day @IRTSystemX
   (b) Date : February 22, 2017
   (c) URL : Watch Video
2. Multicriteria Route Planning
   (a) URL : Download Video
3. Stochastic Route Planning
   (a) URL : Download Video
4. Computing multimodal itineraries
   (a) Event : MIC @SNCF
   (b) Date : November 26, 2015
   (c) URL : Watch Video
5. Rerouting in case of perturbation w.r.t vehicles capacities
   (a) Event : Future @SystemX
   (b) Date : March 10,11 2016
   (c) URL : Watch Video

Abstract: Dynamic Rerouting in Multimodal Transportation Networks

The human mobility is nowadays organized in a multimodal context with more and more complex transport networks. The number of passengers is increasing and new transport modes enter the system day after day simultaneously with new mobility behaviors. As a result, users usually find themselves more confused when choosing between several possibilities to go from one place to their destination. For the sake of helping them to efficiently navigate through this intricate transportation scheme, an efficient Travelers Information System (TIS) has to be built. Thanks to such a system, the transport operators seek not only to provide passengers with optimal itineraries, but also with efficient and reliable solutions in case of disruptions.

In fact, commuters do not only seek short time travels, but they usually consider several other criteria such as comfort and effort. An efficient routing system should therefore take into account the various needs and preferences of each passenger. Besides, transport modes are often prone to delays. Thus, handling uncertainty is also a very important aspect of practical journey planning systems. Moreover, the proposed multimodal routes should not only be feasible in a static case, but also robust against the dynamic and stochastic variations of the transport system. Furthermore, crucial constraints should be taken into account such as the capacity limitation of vehicles and the time complexity of the developed routing algorithms.

The main objective of this thesis is to propose a formulation that adequately allows representing a multimodal network. Based on our formulation, we elaborate several efficient routing algorithms. In particular, we focus on solving the Earliest Arrival Problem in a single/multiple criteria context, both in dynamic and stochastic environments. To deal with the real time complexity issue, metaheuristics such as Genetic Algorithms (GA), Variable Neighborhood Search (VNS) and Memetic Algorithms (MA) have been used.

The computational performance of this work has been assessed by developing a real world route planning system, and solving real life itinerary planning problems defined on the transport network of the French Region Île-de-France that includes the city of Paris and its suburbs. The emerging computational results indicate that the numerous basic and complex instances were solved within a reasonable amount of time and the integration of the proposed routing framework as a module of an operational TIS is relevant.

Keywords: Multimodal route planning, genetic algorithms, variable neighborhood search, memetic algorithms, real world application.

SPIM

■ École doctorale SPIM - Université de Technologie Belfort-Montbéliard
F - 90010 Belfort Cedex ■ tél. +33 (0)3 84 58 31 39
■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

utbm université de technologie Belfort-Montbéliard